The Logical Impingement of AI

Luís Moniz Pereira*

Abstract

We address the impingement of AI on logic, by examining the requirements posed on logic by knowledge representation and reasoning issues which AI has addressed. We then outline some of AI's contributions, via Logic Programming, to more dynamic forms of logic, in order to deal with knowledge in flux, namely: incomplete and contradictory information; hypotheses making through abduction; argumentation; diagnosis and debugging; updating; and learning. Along the way we delve into implications for the philosophy of knowledge.

Our motivation is to entice the philosophically inclined readership to the new territories recently landmarked in the logical continent by ongoing research in AI, by providing a bird's eye view of the region. The more technical parts of the exposition aim at an intuitive understanding, by means of simple examples, of the general issues, how they are approached, and surmounted. The full-fledged technicalities are to be found in the references.

1 Introduction

We address the impingement of Artificial Intelligence (AI) on logic, by examining the requirements posed on logic by knowledge representation and reasoning issues which AI has addressed. We then outline some of AI's contributions, via Logic Programming, to more dynamic forms of logic, in order to deal with knowledge in flux, namely: incomplete and contradictory information; hypotheses making through abduction; argumentation; diagnosis and debugging; updating; and learning. Along the way we delve into implications for the philosophy of knowledge.

Our motivation is to entice the philosophically inclined readership to the new territories recently landmarked in the logical continent by ongoing

*Centro de Inteligência Artificial (CENTRIA), Departamento de Informática, Universidade Nova de Lisboa, 2825 Monte da Caparica, Portugal. e-mail: lmp@di.fct.unl.pt http://www-ssdi.di.fct.unl.pt/~lmp/

research in AI, by providing a bird's eye view of the region. The more technical parts of the exposition aim at an intuitive understanding, by means of simple examples, of the general issues, how they are approached, and surmounted. The full-fledged technicalities are to be found in the references.

The structure of the paper is as follows: The next section recapitulates and emphasizes the unique rôle of logic as a human knowledge representation and reasoning tool. The section that follows prolongs the one before to stress the benefit of mechanizing logic on the computer. Subsequently, the different encroachments of AI on logic are listed, and the last one, a framework whose fresh import for philosophy is underscored, is selected for further analysis. In the sequel, some of the Logic Programming tools bearing on the proposed framework are brought out. Afterwards, the use of such tools for diagnosing a logical theory is outlined. In the section following that a dynamical logic framework is sketched, and then elaborated upon in the ensuing section summarily describing the MENTAL project. Finally, there are Concluding Remarks and a challenge for philosophers.

2 Evolution and Reasoning

Evolution has provided humans with symbolic thought, and symbolic language communication abilities.

Objective common knowledge requires thought to follow abstract, content independent rules of reasoning and argumentation, which must not be entirely subjective, on pain of making precise communication and collective rational endeavour impossible. Such rules have become ingrained in human thought, and hold an enormous joint survival value[1].

In human cognitive evolution, both mimetic knowledge (such as that inherent in reality simulating maps and models), and imitation knowledge (such as that present in ritual observation, or in artifact reproduction), were essential first steps towards socially situated, joint rule following behaviour, required by, say, hunting plans. Subsequently, throughout human cultures, abstract rule following social games emerged.

Game rules encapsulate concrete situation defining patterns, and concrete situation-action-situation causal sequencing, which mirrors causality-obeying physical reality. From games, further abstraction ensued, and there finally emerged the notions of situation-defining concepts, of general rules of thought and their chaining, and of legitimate argument and counter-

[1] Cf. for instance [6][7][16].

argument moves. Together they compose a cognitive meta-game.[2]

The pervasiveness of logic for knowledge representation and reasoning, a veritable *lingua franca* across human languages and cultures, rests on its ability to actually foster rational understanding and common objectivity. Crucially, objective knowledge evolution dynamics itself, whether individual or plural, follows ratiocination patterns and laws.

Moreover, and more recently, the very same rules of reasoning can and are employed to reason about reasoning. Though some reasoning methods are well known, some are still unconcious but, like the rules of grammar, can be discovered through research. Furthermore, new reasoning methods can and have been invented and perfected throughout human history. Examples of these are transfinite induction, *reductio ad absurdum* (proof by contradiction), recursion, abduction, and contradiction removal, to name a few.

New purported reasoning methods may be disputed, just like any specific train of reasoning can. But reasoning can only be disputed by further reasoning, if any consensus is to be found![3] Some argue that scientific and philosophical discussion is necessarily tantamount to a culture sensitive, and culturally relative, persuasive informal *ad hoc* argumentation, allied to anything goes rethoric[4]. They ignore that argumentation is itself a form of reasoning which has been made the subject of logical formalization, and are oblivious to that rethoric may be fine for preachers, but is not conducive to the two-sided communication required to reach common agreement in all the rigorous scientific praxis that lead to cumulative knowledge.

3 Logic and the Computer

Logic, we have seen, has arisen as the content independent formulation of the laws of thought. Predicates express whatever conceptual relations we may wish, about the external and internal worlds. These predicate building blocks can then be combined into formulas, by means of quantifiers and logical connectives, manipulatable according to inference rules characterizing divers forms of reasoning.

Logic is capable of articulating an extensional behaviouristic view of predicates, seen as black boxes or input/output relations, simultaneously with an intensional view of predicates, seen as functionally decomposable

[2] Cf.[9].
[3] Cf. [14]
[4] Cf. [8].

into other predicates, like black boxes wired inside black boxes[5].

Since the language of logic is symbolic and its rules content-independent, its workings can be specified by general, abstract, rule following procedures. These, in turn, can be programmed on the computer. Through their mechanization, logical theories and reasoning forms attain, for the first time, an *in vitro* existence, an ability and availability for repeatable execution, independently of any human mind hardware support[6].

The two views above, extensional and intensional, are reconciled in the computer by insisting that the declarative semantics, the "what" is to be computed, be equivalent to the procedural computational semantics, the "how" it is computed. This is the cornerstone of the Logic Programming paradigm, of which more later.

4 Philosophy and AI

Philosophy has been a cradle for rational thought throughout human intellectual history. Its study of language fostered the refinement of logic as an abstraction of natural language and as a tool of linguistic inquiry. The investigations on the foundations of mathematics too assigned to logic a meta-instrumental rôle that would deploy it as the general purpose symbol manipulating tool *par excellence*. Computer science as well has adopted logic as its general foundational tool, while AI has made viable the proposition of turning it into a *bona fide* computer programming language.

At the same time, AI has developed logic beyond the confines of monotonic cumulativity, typical of the precise, complete, endurable, condensed, and closed mathematical domains, in order to open it up to the non-monotonic real world domain of imprecise, incomplete, contradictory, arguable, revisable, distributed, and evolving knowledge. In short, AI has added dynamics to erstwhile statics.

In AI, as with any science, common logic has an important rôle in it. Indeed, any science, implicitly or explicitly, presses logic and reasoning into its service, to build up arguments, counter-arguments, and to settle disputes. But a greater rôle can be expected of AI in logic!

- First, AI means to mechanize logic, it being such a core tool for so many rational activities.

[5] Cf. [13].
[6] Cf. [15].

- Second, AI intends to make explicit, and well-defined, the unconscious logic we use, and put it to objective test by automating it on the computer.
- Third, AI employs logic as a generic language of communication, of knowledge and of procedures, between humans and computers, and among computers themselves.
- Fourth, in AI, even when procedures and devices are not implemented using logic directly, logic can play the rôle of a precise specification language for their requirements, and of a formalism with which to study their semantical properties.
- Fifth, AI has contributed significantly to the phrasing and examination of the problem of identifying limits to symbolic reasoning embodied in computers, and whether these limits apply to humans as well.
- Sixth and final rôle, AI has helped researchers explore new reasoning issues and methods, and to combine disparate reasoning modalities into a uniform unified framework, so as to deal with incomplete, imprecise, contradictory, and changing information.

This last rôle, in contrast to its predecessor, which has been much pampered by philosophers, has hardly caught their attention yet. The latter-day epigenics of logic it announces, and its computational scaffolding, point to an innovative rôle for the logic enterprise. To wit, that of supporting, justifying, and mechanizing the dynamics of the abstract epistemic and argumentative processes subjacent to scientific endeavour itself. And this, I believe, is good news for philosophers of science.

Consequently, it is to that that we now turn, in the hope that philosophers there will be who shall help it mature into a full-blown computational epistemology.

5 Logical Tools of AI

Classical logic has been developed to study well-defined, consistent, and unchanging mathematical objects. It thereby acquired a static character. AI needs to deal with knowledge in flux, and less than perfect conditions, by means of more dynamic forms of logic. Much of this has been the focus of research in Logic Programming, a field of AI which uses logic directly as a

programming language[7], and provides specific implementation methods and efficient working systems to do so[8]. Logic Programming is also much used as a staple implementation vehicle for other AI approaches to logic.

What are the issues and requirements of the dynamics of logic which AI, namely via Logic Programming, has contributed to solve?[9]

5.1 Logic programming

There is a close connection between logical implication and physical causality. After all, logic allows us to model the goings on in the world. This is even more so in the case of Logic Programming, where all true conclusions must be supported, "caused", by some premises, and where implication is unidirectional, i.e. not contrapositive: "causes" do not run backwards. Horn clause notation is used to express this directionality.

To produce a result or conclusion C we need a conjunction of enough positive conditions P to sustain it, conjoined by ',' to the absence or negation of a conjunction $\neg N$ of all the negative conditions that would prevent it, given P:

$$C \leftarrow P, \neg N$$

If there is more than one way to obtain C, then we have several rules of this form:

$$C_i \leftarrow P_i, \neg N_i$$

If our information about each rule, and about the whole set of rules for C, is complete, then we would have:

$$C \longleftrightarrow \bigvee_i P_i, \neg N_i$$

But what if we don't? First, we might not have enough information about each $\neg N_i$. We might conceivably know about each required P_i, without at least one of which C can never be achieved. But regarding the $\neg N_i$, i.e. all the conditions which, if present, would prevent us from concluding C in the presence of each P_i, that's more difficult.

[7] Cf. [11] for the foundations and [3] for a recent state-of-the art.

[8] For the most advanced, incorporating more recent theoretical developments, see the XSB system at:

//www.cs.sunysb.edu/~sbprolog/xsb-page.html

[9] For a detailed technical treatment of the ensuing issues cf. our book[3], and references therein.

5.2 Open worlds

Too many things can go wrong in an open non-mathematical world, some of which we don't even suspect. For all we know, some bomb might go off that destroys the whole physical setup we're trying to model logically (at a safe distance). Must we model that too? In the real world, any setting is overly complex enough for us to define it exhaustively each time. We have to allow for unforeseen exceptions to occur, based on new incoming information. Thus, instead of having to make sure or prove that some condition N_i is not present, we can assume it is not, on condition that we are prepared to accept subsequent information to the contrary. I.e. we may assume a more general rule than warranted, but must henceforth be prepared to deal with arising exceptions.

Take for example, with the obvious reading, this piece of knowledge, again in Horn clause form:

$$faithful(H, K) \leftarrow married(H, K), \neg lover(H, L)$$

We don't normally have explicit information about who is or is not the lover of whom, though that kind of information may arrive unexpectedly, especially of presidents[10]. Thus, we can code our knowledge on presumption that if someone H is married to K, and there is no information or basis to conclude $lover(H, L)$, then H is faithful to K:

$$faithful(H, K) \leftarrow married(H, K), not\ lover(H, L)$$

This is expressible via so-called default negation $not\ P$, which may be read as "P is not provable". That is, we no longer require proof that $\neg lover(H, L)$ for any L, given some H, but will assume so unless we can establish $lover(H, L)$ for some L given H. In other words, if I have no evidence to conclude $lover(H, L)$ for some L given H , I can assume it false for all L given H.

5.3 Defeasible Assumptions

Default negation was introduced by AI researchers via Logic Programming, and may be read, interchangeably, as "P is not provable", or "the falsity of P is assumable", or "the falsity of P is abducible", or "there is no evidence for P", or "there is no argument for P". Default negation allows us to deal with

[10] At the time of writing I was in the USA, and the american president was being harassed about a purported love affair.

lack of information, a common situation in the real world. It introduces non-monotonicity into knowledge representation. If later we're informed about a pair of lovers, we must then go back on our previous conclusion about whatever faithfulness we had presumed. Indeed, conclusions might not be solid because the rules leading to them may be defeasible. Legal texts, regulations, and courts employ this form of negation abundantly, as they perforce deal with open worlds, though it had not before been formalized in logic, till AI discovered its need for it too.

5.4 Closed World Assumption

Mark that *not* should grant positive and negative information equal standing. That is, we should be able to write:

$$\neg faithful(H, K) \leftarrow married(H, K), not\ \neg lover(H, L)$$

to model instead a world where people are unfaithful by default or custom, and where it is required to explicitly prove that someone does not take any lover before we conclude that person not unfaithful. The issue arises because often, particularly in data bases and knowledge bases, the Closed World Assumption (CWA) in enforced. That is, the CWA obtains when we presume the database or knowledge base to have all the pertinent positive information, so that only what is explicitly stated, or explicitly derivable from it, is true; otherwise, it is presumed false. But how could it, in general, have all the pertinent information in a changing world?

Information is normally expressed positively, by dint of mental and linguistic economics. So, by CWA, the absent, non explicitly obtainable information is usually the negation of positive information. Which means, when no information is available about lovers, that $\neg lover(H, L)$ is true by CWA, whereas $lover(H, L)$ is not. This asymmetry is undesirable, inasmuch predicate names are purely conventional and we could, as justifiably, have come up with positive predicate $non - lover(H, L)$ and its negation $\neg non - lover(H, L)$ to model our knowledge. A logical language should be indifferent with respect to such knowledge representation accidents.

5.5 Explicit Negation

In addition, when we have no factual or derivable either positive or negative information we'd like to be able to say that both are false epistemically, i.e. from the "knowledge we possess" point of view. Accordingly, the excluded

middle postulate is unacceptable because some predication and its negation may be false simultaneously.

The CWA and the symmetry and epistemologic requisites, can be reconciled by reading '$\neg$' above not as classical negation, which complies with the excluded middle postulate stating that any predication is either true or false, but as yet a new form of negation, dubbed in Logic Programming "explicit negation"[11] (which ignores the excluded middle provision). Now we can state the CWA for just those predicates P or $\neg Q$ we wish, simply by writing:

$$\neg P \leftarrow not\ P \quad \text{or} \quad Q \leftarrow not\ \neg Q$$

Alternatively, the use of a $not\ P$ or of a $not\ \neg Q$ assumption may be made at just those predicate occurrences requiring them, like we did before.

5.6 Revising Assumptions

Let us next examine the need for revising assumptions and introducing a third truth-value, call it "undefined", into our framework.

When we combine the viewpoints of the two above worlds we become confused:

$$faithful(H, K) \leftarrow married(H, K), not\ \neg lover(H, L)$$

$$\neg faithful(H, K) \leftarrow married(H, K), not\ lover(H, L)$$

Assuming $married(H, K)$ for some H and K, it now appears that both $faithful(H, K)$ and $\neg faithful(H, K)$ are contradictorily true. Because we have no evidence for $lover(H, L)$ nor $\neg lover(H, L)$, there simply is no information about them, we make two assumptions about their falsity. But when an assumption leads to contradiction one should retract it. It is the venerable principle of *reductio ad absurdum, or* "reasoning by contradiction". In our case, two assumptions led to the contradiction. Which shall we retract? They are on equal footing!

5.7 Undefinedness

Given no other, eventually preferential information, we retract both because we cannot decide between them. That is, we assume neither $lover(H, L)$ nor $\neg lover(H, L)$ false. Since neither is provably true also, we make each

[11] Cf. [4].

undefined. I.e. we introduce a third truth-value to better characterize this lack of information about lovers situation, thereby making $faithful(H, K)$ and $\neg faithful(H, K)$ undefined too. This imposition of undefinedness can be achieved simply, by adding to our knowledge:

$$\neg lover(H, L) \leftarrow not\ lover(H, L)$$

$$lover(H, L) \leftarrow not\ \neg lover(H, L)$$

Given no other information, we cannot prove either of $lover(H, L)$ or $\neg lover(H, L)$ true, or false. Any attempt to do it runs into a self-referential circle involving default negation. However, if we hypothesize one true the other perforce becomes false, and vice-versa. These two possible situations are thus not ruled out. But the safest, skeptical, third option is to take no side in this marital dispute, and abstain from believing either.

Indeed, the well-founded semantics of logic programs (WFS) assigns to the literals in the above two clauses the truth value *undefined* in its knowledge skeptical well-founded model, and allows also for the other two, non truth-minimal, more credulous models.

Why can we not simply add:

$$lover(H, L)\ \vee \neg lover(H, L)\ \ ?$$

Because it will not do. We would still not be able to prove either $lover(H, L)$ or $\neg lover(H, L)$ definitely true, and so the contradiction would ensue all the same!

When dealing with non-provability one really needs a third truth-value to express our epistemic inability to come up with information. Even assuming the world is ontologically 2-valued, our access to information coded about it might be 3-valued in general. Besides, the world may very well not be 2-valued, or, in any case, not capturable in a 2-valued way. Can we say of some particle/wave that it is either here or not here? Is it really either here or not here, though we cannot say it? The inherent dispute apparently cannot be settled experimentally.

In any case we are in wont of a third logical value for other reasons. As we build up our real world imperfect knowledge base, we may very well create, unwittingly and unawares, circular dependencies as above. For example, the Legislator may well enact conflicting, circular, laws. Still, we want to be able to carry on reasoning, whether or not such circularities legitimally express

what they model. And when they do not, we want to detect and function with them rather than throwing away the baby with the bath water.

But undefinedness may also be legitimally pressed into service to express common knowlege, in its variously credulous and skeptical readings. Why, even yesterday I read in a book[12] of the old sailor proverb that "He weathers the storms he cannot avoid, and avoids the storms he cannot weather", i.e.:

$$weather \leftarrow storm,\ not\ avoid$$

$$avoid \leftarrow storm,\ not\ weather$$

What will happen to an experienced sailor in a storm? Will he weather it or avoid it? Three outcomes are possible, including being undecided about the outcome. For us, it means we must weather circular knowledge, not avoid it in the calm waters of the artificial mathematical paradises...

5.8 Logic Programming Expressiveness

The introduction of this 3-valued semantics in logic programming is an important innovation of AI, with a number of consequences. Namely, we have seen, it allows us to be skeptical and suppose no more than is warranted.

Another innovation, to recapitulate, is that the negation $\neg$ used above, called explicit negation, is 3-valued too, and so differs from classical negation. Explicit negation, we've seen, does not conform the principle of excluded middle. Predications and their negations can be both false ("the chair is angry", "the chair is not angry"). In a knowledge base, we may definitely not be able to prove either something or its (explicit) negation, so that both become false by default.

Additionally, the introduction of explicit negation, and in particular its combination with default negation, provides for more expressive knowledge representation. Consider for instance the injunction

$$\neg cross \leftarrow train$$

versus the alternative injunction

$$\neg cross \leftarrow not\ \neg train$$

The latter is clearly a safer option. According to it you must actually prove that a train is not coming before $\neg cross$ becomes false, whereas with

[12] Cf. [5].

the first option $\neg cross$ becomes false solely on the ground that you fail to prove a train is coming!

Yet another innovation is that the implicational arrow $\leftarrow$, when combined with default or explicit negation, is not material implication. Contrapositives are not countenanced. Consider:

$$\begin{array}{rcl} A & \leftarrow & B \\ & & \neg A \end{array}$$

$\neg B$ does not follow, for there is no presumption of the contrapositive $\neg B \leftarrow \neg A$. Whenever contrapositives are desired they must be explicitly added. Instead, $\leftarrow$ should be seen as an inference rule, which can be used "bottom-up" to conclude A given B, or "top-down", like an invoked procedure, to try and prove the body B in order to prove the head A. Facts are just procedures with empty body. This inferential reading of clauses as rules makes $\leftarrow$ a directional operator, which makes it rather appropriate to model not only inference but causality as well.

Indeed, the semantics of logic programs emphasize this with their insistance on admitting only minimal models. Minimal models are those for which every positive or explicitly negated literal *true* in the model is supported on some rule whose head or conclusion is the literal, and whose body or set of premises is in turn supported. Thus facts, since they have an empy body, are automatically supported.

Additionally, any positive or any explicitly negated literal is *false* just in case all rules for it have a false body. Consequently, if there are no rules whatsoever for a literal it is automatically false.

All other positive or explicitly negated literals are *undefined* (this can happen only if they are involved in unresolved self-referential loops through default negation). Finally, a default negated literal, *not* P, is *true/ false/ undefined* just in case P is, respectively, *false/ true/ undefined*.

5.9 Updating

Last but not least, work in logic programming has concerned itself with the updating of a knowledge base by another one. This notion of knowledge updating, as opposed to that of simple fact updating, opens up another dimension to the dynamics of logic, in contradistinction to the statics of the logic of old. Given an existing knowledge base, containing facts and rules, and some new knowledge with facts and rules as well, possibly contradicting the previous knowledge when added to it, what is the resulting updated knowledge base? Can this process be iterated?

Recent work has shown how this can be achieved with generality[13]. Application areas in point are when the two knowledge bases consist in pieces of legislation, or regulations, or safety procedures, or rules of robot conduct.

6 Logical Theory Diagnosis

Many examples exist of the use of default negation, explicit negation, contradiction removal, and 3-valuedness, made possible by research in Logic Programming, namely in the areas of abduction, argumentation, belief revision, knowledge upating, learning, and diagnosis[14]. Let us illustrate the latter, i.e. the diagnosis or debugging of a generic knowledge base. Its connections with the other aforementioned areas allow us to brush on them too.

Suppose we have the following rules, expressing a logic program that allows us to compute predicate C on the basis of predicates P and N:

$$C(X) \leftarrow P(X), \neg N(X)$$

$$\begin{array}{ll} P(a) & \neg N(a) \\ P(b) & \\ P(c) & \neg N(c) \end{array}$$

Let us allow for the chance that the rule for C is conceivably wrong. i.e. that it may have exceptions, by writing it thus:

$$C(X) \leftarrow P(X), \neg N(X),\ \mathit{not}\ \mathit{exception}(C(X))$$

If nothing else is stated about the *exception* predicate, the conclusions of the logic program remain unchanged, so the program may contain these rules from the start, for any predicate we wish.

If we next learn that $C(a)$ is false, we can salvage the rule for C, since it is no longer valid for all $X = a$, simply by adding to our knowledge:

$$\mathit{exception}(C(a))$$

The rule for C will continue to work for the remaining cases, though more exceptions might subsequently be accumulated.

This method takes care of rules that are unsound, i.e. produce wrong results.

[13] Cf. [2].
[14] Cf. [3].

What about the other problematic case, that of incompleteness, i.e. when results that would be correct are missing? Suppose we learn that $C(b)$ should true, but which our rules rule out? Well, it is enough to introduce for every set of rules for a predicate a catch-all rule. In this case:

$$\begin{aligned} C(X) &\leftarrow missing(C(X)) \\ P(X) &\leftarrow missing(P(X)) \\ \neg N(X) &\leftarrow missing(\neg N(X)) \end{aligned}$$

If nothing else is stated about the *missing* predicate, the conclusions of the logic program remain unchanged, so the program may contain these rules from the start. Now, to make $C(b)$ true, it now suffices now to abduce, or adopt, one of two hypotheses: that $missing(C(b))$ or that $missing(\neg N(b))$ is true.

This method takes care of rules that are incomplete, i.e. fail to produce results.

We have achieved generality. By making the two predicates, $exception(_)$ and $missing(_)$, abducible ones, we can diagnose and debug a knowledge base expressed in logic.

Abduction is a well known reasoning process by which one can adopt hypotheses to the effect that some predicate instances (or their negations) are true in order to prove some conclusion. It's reasoning from goals to requirements, and has been extensively studied in the logic programming context[15]. Of course, the assumptions or hypotheses so adopted may later prove erroneous, if and when they lead to contradiction. We have then to make provision for the application of a contradiction removal process based on revising assumptions, possibly adopting of some other assumptions instead. This too has been extensively studied in AI and in the logic programming setting, and as a result automated reasoning systems have been made available which do that work for us[16].

7 A Dynamical Logic Framework

It is not too difficult to imagine how a combined process of rule generation, of systematic diagnosis, and of rule revision by updating, can be used to achieve automated theory learning, in an integrated way, within the uniform setting of logic programming.

[15] For an overview, cf.[10].

[16] Cf.[1].

To initiate the learning, one starts with some fixed, already acquired, background knowledge in rule form, i.e. a theory, and a rule generator to add to it new purported knowledge, in order to explain abductively known observations, whether positive or negative, in the form of facts and explicitly negated facts.

The goal is to generate rules that define a positive concept as well as its negated concept, so that they cover all known observation instances. This automatic generation of new rules is subjected to a pre-defined bias, i.e. only some rule forms, and predicates comprising them, are allowed in the generation process. Newly generated rules may contradict one another, on some of the observation instances, and so they must be subjected to a diagnosis, to identify alternative possible minimal revisions.

To decide which revision to adopt next, desirable new possible observations are conceived of, with respect to the ongoing available theory, and whose results, if known, would allow to decide among the competing revisions. The results of these so-called crucial observations are then obtained, either by the program soliciting them, or by a planning process subsequently executed to carry out the actions leading to the observation results

Once the desired revisions are selected on the basis of the results, and of also programmed preference criteria, the revisions are enacted by an update procedure. Note that revised rules can themselves be subjected to later revisions if needed.

Indeed, the whole process will be iterated on the basis of new incoming knowledge, or by knowledge confrontation of among differently evolved automated theories, with distinct backgrounds, biases, rule generators, diagnosers, revisors, preferences, planners, observations, and updating procedures, comprising a rational agent.

Agent epistemic confrontation relies on argumentation and mutual debugging. Besides the legislative and legal domain, the province of scientific discussion too relies on such procedures, and can benefit from their automation.

Tackling argumentation involves a set of tools similar to those of diagnosis and debugging. Arguments can attack another argument's assumptions either directly, by proving the negation of an assumption, or indirectly, by contradicting a conclusion of other argument which rests on its assumptions. However, such attacks by an argument upon another can, in turn, be counter-attacked in the same way, and may in response counter-counter-attack, etc. Logic programming has shown how this process can be studied and conclusions drawn about competing mutually contradictory arguments, and how they each can be revised to reach agreement.

8 The MENTAL project

The above sketch of a dynamical logical framework is being made concrete within the context of our ongoing MENTAL agents project[17], at the Centro de Inteligência Artificial, Universidade Nova de Lisboa.

In both academia and industry it is increasingly felt that intelligent agents will be a key technology as computing systems become ever more distributed, interconnected, and open. In such environments, the ability of agents to autonomously plan and pursue their actions and goals, to cooperate, coordinate, and negotiate with others, and to respond flexibly and intelligently to dynamic and unpredictable situations will lead to significant improvements in the quality and sophistication of the software systems that can be conceived and implemented, and the application areas and problems which can then be addressed.

The aim of the MENTAL project is that of establishing, on a sound theoretical basis, the design of an overall architecture for mental agents (i.e. comprising knowledge, beliefs, and intentions) based on, and building on the strengths of logic programming.

The agents will share a common changeable environment, itself modeled in logic programming. The whole framework is being conceptualized in logic programming, and experimented with by means of distributed logic programming. The use of agents allows for information to be distributed, and shared, only as the need arises. Thus knowledge manipulation complexity can be reduced. Efficiency can be gained too by executing agents concurrently.

An agent must be able to manage its knowledge, beliefs, intentions (goals), and plans, as it receives new information and instructions, and to react to changing conditions in the environment. It must also be capable of interacting with other agents, by exchanging knowledge and beliefs, as well as of reacting to other agents' requests. Any two such agents will be able to cooperate either by diagnosing errors or missing information in each others' information base, or to cooperate in the use of common resources, in avoiding undesired mutual interference in their plans, and in joint planning to achieve a common goal.

Every agent is composed of specialized, possibly concurrent, function related subagents, that execute the various goals and instructions assigned to it by the agent. Examples of such subagents are those implementing the reactive, reasoning, planning, belief revision, argumentation, explanation,

[17] PRAXIS XXI project MENTAL (2/2.1/TIT/1593/95).

learning, dialogue management, information gathering, preference evaluation, strategy, and diagnosis functionalities. The subagents are coordinated by a meta-level layer that ensures internal task distribution and planning, subagent communication, final decisions, and all interaction with the outside, i.e. with both the environment and other agents, including communication, interrupt handling, requests, and observations.

Although each agent has a meta-level coordination, a collection of interacting agents has no such abode, and their collective behaviour will manifest difficultly foreseeable (emergent) properties. Because knowledge and belief are in general incomplete, contradictory, and error prone — and all the more so in the multi-agent setting — we make use of semantics, procedures, and implementations, for dealing with contradictory, incomplete, erroneous, imperfect, vague and default information, abduction, belief revision, debugging, and argumentation.

We propose to attain reinforcement learning by using belief revision techniques for achieving an effect similar to back propagation. Such learning will allow the evolving of the strength of evidence attached to an agent's knowledge and beliefs, as a result of their contribution to correct or incorrect conclusions. Furthermore, agents may compare and combine their degrees of evidence, either to argue, to partake of information, or to reach consensus. The use of genetic algorithms for evolving belief "genes" (or "memes") is also a technique worth exploring for learning, and it opens up a new emergent field: that of linking the genetic algorithm approach to logic based knowledge evolution in an agent. Moreover, the exchange of genetic material will enable agents to cross-fertilize experiences.

The use of logic programming for this endeavour is justified on the basis of it providing a rigorous single encompassing theoretical basis for the aforesaid topics, as well as an implementation vehicle for parallel and distributed processing. Additionally, logic programming provides a formal high level flexible instrument for the rigorous specification and experimentation with computational designs, making it extremely useful for prototyping, even when other, possibly lower level, target implementation languages are envisaged.

9 Concluding Remarks

How can the general reasoning issues explored here be approached, within a uniform framework, except in the context of logic? I submit that logic is the only conceivable venue for such an enterprise. And must we not do it? Must

we not mechanize reasoning procedures so that robots and computers can perform, for us or with us, the sometimes tedious sometimes overly complex work involved in them? Surely, only so persisting will permit the grappling of the evermore demanding reasoning problems that face us.

I hope to have convinced you that AI, most especially through Logic Programming, will continue to accomplish a good deal in identifying, formalizing, and implementing the laws of thought. Most notably, AI has taken on the challenge of opening up logic to the dynamics of knowledge in flux. And in so doing, it has been progressively meeting our expections and requirements. Continuing this work is essential if we are to cope, albeit with the aid of computers, with the challenges of evermore accumulated and distributed knowledge, in a changing world.

Participation by philosophers can take the form of partaking in the development of the conceptual underpinnings of the required computer implemented logic tools, and of employing such tools for the computer functional rational reconstruction of historical epistemological argument[18], and of scientific theory evolution duplication, by automated learning. Taxonomic classification of natural kinds would be a good starting point.

10 Acknowledgements

This work was partially supported by PRAXIS XXI project MENTAL (2/2.1 /TIT/1593/95), and a NATO scholarship while the author was on sabbatical leave at the Department of Computer Science, University of California, Riverside. Thanks are due to Luís Correia, to João Leite, and to Halina Przymusinska for their insightful comments.

References

[1] J.J.Alferes, C.Damásio and L.M.Pereira, "A logic programming system for non-monotonic reasoning". Journal of Automated Reasoning, 14:93-147, 1995.

[2] José Júlio Alferes, João Leite, Luís Moniz Pereira, Halina Przymusinska and Teodor C. Przymusinski, "Dynamic Logic Programming", Int. Conf. on Knowledge Representation and Reasoning (KR'98), Trento, Italy. Stuart Shapiro (ed.), Morgan Kaufmann, 1998.

[18] As can be found for instance in .

[3] José Júlio Alferes and Luís Moniz Pereira, "Reasoning with Logic Programming", Lecture Notes in Artificial Intelligence, vol. 1111, Springer, 1996.

[4] J.J. Alferes, L.M.Pereira and T.C.Przymusinski, "'Classical' Negation in Nonmonotonic Reasoning and Logic Programming", Journal of Automated Reasoning, 20:107-142, 1998.

[5] John Barth, "On with the story", page 55, Back Bay Books, Little Brown and Company, 1996.

[6] Terrence Deacon, "The Symbolic Species", W.W. Norton, 1997.

[7] Merlin Donald, "Origins of the Human Mind", Harvard University Press, 1991.

[8] Paul R. Gross and Norman Levitt, "Higher Superstition", The Johns Hopkins University Press, 1994.

[9] John Holland, "Emergence", Addison-Wesley, 1998.

[10] A.C.Kakas, R.A.Kowalski and F.Toni, "Abductive Logic Programming", Journal of Logic and Computation, 2:719-770, 1993.

[11] Robert Kowalski, "Logic for Problem Solving", North-Holland, 1979.

[12] Imre Lakatos, "Proofs and Refutations: the Logic of Mathematical Discovery", Worrall, J. and Zahar, E.G. (eds), Cambridge University Press, 1976.

[13] Luís Monteiro and Luís Moniz Pereira, "Aspectos Cibernéticos da Epistemologia", in "Novas Perspectivas das Ciências do Homem", pp. 13-27, Biblioteca de Ciências Humanas 4, Editorial Presença, Lisboa, 1970.

[14] Thomas Nagel, "The Last Word", Oxford University Press, 1997.

[15] Luís Moniz Pereira, "Inteligência Artificial: mito e ciência", in Revista Colóquio-Ciências, 3:1-13, October 1988, Fundação Calouste Gulbenkian, Lisboa.

[16] Steven Pinker, "How the mind works", W.W.Norton, 1997.