[For Evolving Knowledge Conference. Reading University Sept 1989]
IN:
Evolving Knowledge in Natural Science and Artificial Intelligence,
eds J.E.Tiles, G.T.McKee, G.C.Dean,
London: Pitman, 1990
Comments to A.Sloman@cs.bham.ac.uk

Must Intelligent Systems Be Scruffy?

Aaron Sloman
School Of Cognitive and Computing Sciences
University of Sussex
Brighton
[Now School of Computer Science, the University of Birmingham]

Introduction: Neats vs Scruffies

You would probably be very unhappy about air travel if you were told that landing was always under the control of programs that were so complex and messy that nobody really understood how they worked, and nobody had ever proved that they would do something sensible in all the situations that could possibly arise. Would you be equally unhappy to learn that your life was in the hands (excuse the metaphor) of a human brain?

There has been a long-standing opposition within AI between "neats" and "scruffies" (I think the terms were first invented in the late 70s by Roger Schank and/or Bob Abelson at Yale University).

The neats regard it as a disgrace that many AI programs are complex, ill-structured, and so hard to understand that it is not possible to explain or predict their behaviour, let alone prove that they do what they are intended to do. John McCarthy in a televised debate in 1972 once complained about the "Look Ma no hands!" approach. Similarly, Carl Hewitt, complained around the same time, in seminars, about the "Hairy kludge (pronounced klooge) a month" approach to software development. (His "actor" system was going to be a partial solution to this.)

The scruffies regard messy complexity as inevitable in intelligent systems and point to the failure so far of all attempts to find workable clear and general mechanisms, or mathematical solutions to any important AI problems. There are nice ideas in the General Problem Solver, logical theorem provers, and suchlike but when confronted with non-toy problems they normally get bogged down in combinatorial explosions. Messy complexity, according to scruffies, lies in the nature of problem domains (e.g. our physical environment) and only by using large numbers of ad-hoc special-purpose rules or heuristics, and specially tailored representational devices can problems be solved in a reasonable time.

Moreover, the scruffies tend to complain that the neats spend so much time worrying about form (e.g. defining formal syntax and semantics for their notations) that they ignore content (e.g. what information about the world an intelligent system really needs). They sometimes even liken the "theorem envy" of those who hanker after mathematical rigour to the kind of arid statistics-based research in psychology and the social sciences that uses sophisticated mathematical techniques (and much computer power) with little or no increase in understanding of any interesting phenomenon.

Recently the conflict has taken on a new twist, with the rise of connectionism. Here we have a branch of AI (yes, it is part of AI, not a new rival discipline), that is heavily mathematics-based, yet, although the general principles on which a particular network learns during its training period may be well understood, the operation of the final system when applied to real tasks generally depends on a totally opaque network of connections between processing units and weights that modify their influence on one another. Only in relatively simple cases is it possible to interrogate the "hidden layers" of a neural net and find out what they are doing or why it works. So it looks increasingly as if systems based on neural nets will, after training, be even more inscrutable than AI programs: we'll just have to trust that it is safe to extrapolate from the test cases where they appear to perform well -- exactly as we do with people to whom we give responsibility. Alas!

This is not unlike the situation in weather forecasting, where the general physical principles underlying the weather are well understood but the boundary conditions are impossible to measure and represent with comlete accuracy and their large scale effects are impossible to compute within acceptable margins of error: small errors in initial inputs to equations can produce unpredictably large errors in their solutions.

Anyhow, quite apart from the implications of connectionist models of neural nets there are more general things to be said about the neat/scruffy conflict. I shall try to separate different strands in the conflict and will conclude that there are good reasons why intelligent systems performing in real time in a real world, or even in some purely formal problem domains, will have to be scruffy. Still, wherever possible we should use neat theories to guide our designs and explanations. In particular, scruffy systems should be based on neat designs.

We can contrast the theories about the world and how to relate to it that are used by working intelligent systems with theories about the design of such systems. I'll call the former object-level-theories and the latter meta-level-theories. My point is that we should strive to make meta-level-theories as neat as possible, including the theories that explain why object-level-theories are bound to be scruffy.

The scope of AI

One source of optimism about the possibility of neat designs is an excessively narrow view of the scope of AI. For example there are some who tend to think of AI as synonymous with the field of expert systems, and it is true that for many (though not all) expert systems the problem domain is sufficiently restricted that complete analysis is possible. I prefer a definition that reflects the range of work reported at AI conferences, in AI journals, and the interests and activities of some of the leading practitioners, including founders of the subject. From this viewpoint AI is a very general investigation of the nature of intelligence and the principles and mechanisms required for understanding or replicating it, and this includes topics like vision and motor control which are important for complete intelligent systems embodied in the physical world.

Like all scientific disciplines AI thus construed has three main types of goal, empirical, theoretical, and practical, encompassing:

(a) empirical study and modelling of existing intelligent systems (including human beings and other animals);

(b) theoretical analysis and exploration of possible intelligent systems and possible mechanisms, architectures or representations usable by such systems, and using the results to explain the empirical phenomena observed under (a);

(c) solving practical problems in the light of (a) and (b), namely:

(c.1) using the understanding gained in (b) to deal with problems of existing intelligent systems (e.g. problems of human learning or emotional difficulties)

(c.2) using the theoretical understanding to design new useful intelligent machines

(c.3) using the theoretical understanding to design new machines that may not in themselves be intelligent but nevertheless usefully complement human intelligence.

In the course of these activities AI generates new sub-problems, and these lead to new concepts, new formalisms, and new techniques. On this general definition there is little difference between AI and Cognitive Science, though in fact many of the people who call themselves cognitive scientists are ignorant of the contents of most AI research and its relevance to their own work.

Much of the theoretical work that falls under this general view of AI includes understanding design trade-offs relevant to intelligent systems. For example, there are trade-offs between efficiency and generality, between complexity of design and the ability to degrade gracefully when things go wrong. The rest of this paper is concerned with trade-offs that confront an intelligent system that has to solve problems within constraints of time and memory.

Bow to the inevitable: why scruffiness is unavoidable

The main reason why scruffiness is inevitable in intelligent systems can be summarised thus: in complex domains, neatness is defeated by combinatorial explosions, unless you have an infinitely fast processor or all eternity to wait for solutions.

The argument runs as follows: except in limited cases (to be discussed below) problem domains have a structure that requires solutions to be found by exploration of branching paths in a search space. The number of possible branches will be (at least) an exponential function of the depth of path to solution. E.g. if N steps are required and the number of possible branches per step is as low as 2, the number of possible paths will be 2**N. So even if only 50 steps are required, and they can be explored at the rate of a million a second they will need about 35 years for an exhaustive exploration. A mere 70 steps would require about 37 million years. That indicates how exponential functions explode: adding a mere 20 of whatever the cost is an exponential function of multiplies the time or space requirements by about a million for a branching factor of 2 or over 3000,000,000 for a branching factor of 3. These functions grow so fast that they are not going to be helped much by parallel implementations.

For problem domains that have this character it is very often the case that neat, general, sound and complete algorithms work only on small problems: they do not scale up because they have to do so much detailed analysis of cases that they explode combinatorially.

Non-explosive domains

Of course, not all search problems are like that. If you have a very large array of names alphabetically ordered and want to search for a particular name, you can (given suitable addressing mechanisms) use the "binary chop" method, i.e. look at the middle, then decide whether to go left or right, then look in the middle of the remaining region, etc. With this method the relation between problem size and solution time is the reverse of exponential, it's logarithmic. I.e. the size of the problem is an exponential function of the time it takes to solve it rather than the other way round. So even if the problem (the list of names) is made a million times longer, you will typically need only about 20 more steps to find a name. That's how telephone directories work.

There are other problems that have tractable complexity. For example if you are trying to find some combination of N values of variables that solve a problem, instead of searching through all N factorial possible combinations you may be able to structure the problem as a search in a relatively homogeneous N-dimensional vector space, where there is at any point a function that tells you in which direction to move towards the solution point. Then you can home in on the solution fairly directly by moving along the line of steepest ascent to the goal, e.g. using relaxation techniques.

For more complex problems that can be represented in an N dimensional vector space, for instance pattern recognition problems, it is well known that this simple hill-climbing strategy won't work, e.g. because you can end up on a local hillock or get lost rambling aimlessly on a plateau with no clear direction of steepest ascent. However, some kinds of neural nets are capable of being trained on a particular class of problems so that they transform such a space into a different space in which solutions are found very quickly by going along the direction of steepest ascent or descent, though there is still much to be learnt about the classes of problems for which this works, and under what conditions the training methods will produce complete and sound problem solvers.

Unfortunately, it is far from obvious that all the problems that an intelligent agent needs to deal with can be expressed in a form that is amenable to these solution methods. For example for some problems (e.g. searching for proofs in pure number theory) the structure of the domain makes it impossible either to set bounds to the length of proofs or to avoid branching searches for proofs. This makes it impossible to map the search space into a finite dimensional vector space amenable to relaxation or neural net techniques that home in on solutions without combinatorial searching: we are again confronted by these horrendous exponential functions.

This is why nobody tries to solve problems in arithmetic by simply remembering Peano's five axioms and searching for a logical proof based on those axioms. Instead we memorise all kinds of strictly redundant facts derivable from those axioms, including multiplication tables and lots of rules of thumb for solving particular problems (e.g. you can tell if an integer is divisible by five by looking at its last digit in the decimal representation, or whether it is divisible by three by seeing whether the sum of its decimal digits is divisible by three, etc.).

These stored "lemmas" (previously proved results) are facts that are strictly redundant but have been found to be useful for classes of problems that occur frequently. For other problems combinatorial search may still be required.

Of course, if you depend so much on stored facts you have a new problem which is to find a way of organising them so that you can tell which one, if any, is relevant to a particular task. There is no infallible way of doing that: discovering relevance can be has hard a problem as any.

So, for the full potentially infinite collection of arithmetical problems, we are about as badly off with the stored intermediate results as we are without them. Fortunately, however, it is an empirical fact that for most people's lives there's only a comparatively small, bounded, variety of arithmetical problems that they ever have to solve, so it pays to learn the ad-hoc collection of facts and tricks that enable such problems to be solved (with or without our understanding why).

Those unfortunates who do research in pure number theory have no such complete problem solving kit, and therefore have to spend their lifetime groping around without such effective aids to rapid solutions, though even they tend to specialise in sub-fields where they can build up expertise in the form an ad-hoc collection of useful heuristics, lemmas, and patterns, for deciding what's worth trying next.

In chess, where, unlike the set of possible proofs in number theory, the move tree is finite, it is nevertheless so large that as regards strategies for deciding in a reasonable time where to move, the tree might as well be infinite. Of course there are some important relatively small subtrees for which winning algorithms have been discovered, and good chess players learn patterns on the board to recognize when they have entered such subtrees, or when they are close to entering them.

So unless someone invents an infinitely fast, infinitely large, processing system, any intelligent agent will have to find short cuts to solving problems, even when dealing with formally specified domains. In fact, short cuts are the key to intelligence: a major component of intelligence is *productive laziness.*

Here's an example of a short cut using a special purpose representation. Suppose you are asked to find all possible ways of selecting values between 1 and 9 inclusive for the variables a to i, satisfying the following conditions.

1. No two variables have the same value.
2. all the following are true:
   $a + b + c = 15$
   $a + d + g = 15$
   $a + e + i = 15$
   $b + e + h = 15$
   $c + f + i = 15$
   $d + e + f = 15$
   $g + e + c = 15$
   $g + h + i = 15$

You can search through all possible ways of assigning the 9 integers to the 9 letters, i.e 9 factorial or 362880 combinations. However, a little representational and conceptual creativity can almost completely eliminate the searching required. The trick requires the following observations

(a) The problem is equivalent to mapping the integers 1 to 9 onto locations in a 3 by 3 array of squares labelled thus

a b c
d e f
g h i

such that all rows, columns and diagonals add up to 15.

(b) There are three distinct types of locations in the array, occurring in different numbers of collinear triples:

the centre (e) occurring in four triples,
corners (a, c, g, i) occurring in three
middle edges (b, d, f, h) occurring in two triples.

(c) So there must be three different types of integers between 1 and 9 such that some occur in four, some in three, and some in two triples adding up to 15.

It takes very little work to divide the nine integers into these three categories. After that assigning them to locations in the array so as to satisfy the conditions requires almost no backtrack searching. After one solution is found all the others are generated quickly by using the symmetries that are blindingly obvious in the geometrical representation though not at all obvious in the original formulation of the problem.

This example of productive laziness used the ability to notice a relationship between an arithmetical problem and a geometric structure, as a result of which some additional arithmetical concepts were created (concepts of different kinds of numbers) which can then be used to constrain the assignment of numbers to letters in such a way as to more or less eliminate searching.

This toy example is typical of creative problem solving with much harder problems in mathematics and elsewhere. Often a problem looks either unsolvable, or solvable only by very tedious exhaustive search, until a relationship between domains, or between problems, is noticed that provides a short cut to a solution.

Sometimes discovering such relationships is just a matter of luck. Sometimes it results from a generalisation or modification of a similar relationship that has already been shown to have heuristic power.

However, for many kinds of problems short cuts work if you are lucky but they are not perfect. Moreover, most good short cuts cannot be worked out in advance on the basis of prior analysis of the domain: they have to be discovered by exploration and analysis, like the way in which a great chess player (or mathematician) learns patterns by observing and generalising what turns up in actual experience instead of trying to do everything in advance by applying general inference procedures to the rules of chess (or the axioms of a branch of mathematics).

So even in formal and mathematical domains there are not going to be neat generally applicable powerful methods that can be used to solve all problems in a reasonable time.

The physical (biological, social) world is even harder to deal with

The requirement to discover rules of thumb and short-cuts by exploration and experience is even more pressing in connection with problem solving in the real world (e.g. finding good ways to build houses that are resistant to wind, rain, cold, etc.). There are several reasons why things are harder in the real world, including the following:

a. Unlike a formal system like chess or number theory, no complete body of information is available as a starting point: knowledge is always limited. (In fact we are generally both ignorant about some things and misinformed about others.)

b. The range of possible things to do, and therefore the branching factor in search spaces will be considerably higher than in chess or arithmetic: you can scratch your ear, shout for help, look for a library, try to climb a tree, think of a number, try to remember whether you have been here before, etc., etc. So search spaces are unmanageably large, especially if the task is to find all the solutions, or to find the best one.

c. Things happen at speed in the world, and very often decisions and actions are required by a deadline in the immediate future. This rules out extensive combinatorial searches.

For all these reasons rapidly accessible, rapidly executable rules of thumb have to be derived or learnt by trial and error. Because the world is treacherous and we are not gods, such rules are highly fallible.

In both the real world and formal domains mistakes are possible. A heuristic induced from actual experience can be erroneous because the wrong generalisation was made, even in a mathematical problem domain. Additional problems in the physical world are poor observation or measurements, or a poor set of concepts for dealing with the phenomena - like trying to think about movement of masses without the concept of acceleration. There are no general, provably correct, algorithms for finding powerful new notations and using them to define new concepts. Scientific and cultural developments are slow, erratic, and often painfully wrong. And this is inevitable.

Limits of consistency in intelligent systems

Given that some of your rules, generalizations and facts may be wrong one possible (though not infallible) way of detecting that errors exist might be to check whether they are all mutually consistent. If not, something must be wrong, though consistency does not imply that everything is fine.

Alas, consistency checking is itself subject to combinatorial explosions. Even if all complexity in the knowledge base is due to the use of propositional connectives, and no quantifiers are involved, if there are N atomic propositions a consistency check can take up to 2**N tests, which will be an impossibly large number for a realistic database. Of course, particular cases can be dealt with more quickly, but in general checking consistency in a large knowledge base is an intractable problem.

Moreover, even if you've managed to achieve internal consistency, maintaining it is a new problem if your beliefs are error prone, and you frequently have to modify beliefs. Because the knowledge base has to be redundant for reasons mentioned previously, retracting a belief raises the problem of finding and dealing with other redundant beliefs, plans, rules, etc. that were derived from the one that has been rejected.

You could in principle embed your knowledge base in a reason maintenance system that kept records of all such dependencies, telling you what else must be retracted as a consequence of correcting errors, but again in practice this strategy will be defeated by the sheer volume of dependency relations, in any realistic knowledge base.

So the only *practical* strategy on revising a belief is to use whatever heuristics are available for finding closely related beliefs that also need to be revised, and then hope that if there are any other beliefs that were derived from the erroneous one you will later independently detect their wrongness and correct them before they lead you into trouble. In practice we don't always avoid such trouble, and the arguments presented here suggest that that is not just because human beings are badly designed but because the problem will inhere in any resource-limited system.

Scruffy semantics

There is an additional kind of scruffiness that I think is inevitable in intelligent systems inhabiting a rich, and only partly knowable, world: namely scruffy semantics.

Logicians like to define formalisms whose syntax and semantics are specified precisely.

A definition of the syntax of a formalism starts by assuming some class of structures (e.g. all possible sequences of characters in some alphabet, or all possible networks made of certain kinds of nodes and links) and then giving a rigorous definition of a subset of such structures that are taken to be syntactically well formed (in the formalism being defined). Any structure will either be definitely well formed or not.

Human beings often deploy representational formalisms that are open-ended in ways that are incompatible with this conception of syntax. For example, maps are widely used, but there is no set of rules specifying exactly what is and what is not a legal map. It is up to someone creating a map to decide what will be an effective device for the intended purpose, assuming that the eventual user will be creative and intelligent. As a result there is considerable variation in road maps published by different firms. Some of them use considerable ingenuity in inventing new encodings of information, for example whether a link between a motorway and a lesser road allows both entry to and exit from the motorway, or only one of these -- a problem that used not to arise for more primitive road systems.

Sometimes the conventions used in a map are explicitly specified in a key, but sometimes it is up to the reader to infer them, using general knowledge of the domain and the purposes of the map.

This kind of open-endedness applies to all kinds of pictures, diagrams, charts, tables, models, etc used in human communication. The prior existence of some agreed definition precisely specifying a syntax is the exception rather than the rule.

Moreover, if we were not allowed this syntactic creativity in our communicative devices we would be impoverished not only culturally but also scientifically and technically. Even mathematicians often create new diagrams and notations both when searching for proofs and when communicating concepts and proofs to others.

Of course, natural languages are equally prone to creative extension. When a toddler once said to his parents "Today might be much more hotter than it usually be's", in a discussion of whether to go on a picnic, they understood him perfectly. More generally, the syntax of natural languages is constantly evolving, and evolving simultaneously in different directions in different sub-cultures.

Semantic scruffiness is an even deeper issue than syntactic scruffiness. A formal semantics of the kind defined by Tarski (Tarski 1956) maps well formed expressions in a formalism onto elements or subsets of a previously specified set of objects in a such a way that whether something is or is not so mapped is determined by a collection of recursive rules.

In real life we are not in a position to specify precisely which sets of objects we are dealing with nor the mapping rules.

Instead we muddle on with a collection of relatively ill-specified notations, and employ structures using such notations with a collection of ill specified semantic rules. In particular it is not generally the case that for human language the question whether a situation is or is not correctly described by a sentence has a determinate answer. "That is a giraffe" might be indeterminate when a new kind of animal is discovered that has some of the features previously regarded as essential to being a giraffe, but not all of them. Many scientific advances depend on the introduction of new concepts that are later found to have semantic indeterminateness, for instance the concept of mass in newtonian mechanics. When the indeterminateness is discovered, e.g. because an unanticipated situation arises in which defining criteria generate conflicts, this often catalyses conceptual changes that enable us to extend our concepts and theories to provide us with a deeper more powerful understanding of the world.

A full discussion of this issue is impossible in the space available here. I'll simply baldly assert that for creative intelligent agents groping towards ever increasing knowledge and understanding of an indefinitely rich and complex world it is inevitable that the notations will have both syntactic and semantic indeterminacy of a kind that both limits the applicability of logical and mathematical methods but also provides a stimulus to creative advance.

So various kinds of scruffiness are inevitable

If all this is correct, then over the years any intelligent learner in the real world, as opposed to some ideal world in the mind of wishful logicians, will build up an ever increasing store of ill-defined information, much of it inevitably redundant, much of it inevitably wrong, and with no hope of keeping it all internally consistent. I.e. real intelligent agents will be irretrievably scruffy.

I have argued elsewhere (Sloman 1987) that there will be additional kinds of scruffiness to do with the requirements of control systems in an agent that has multiple independent sources of motivation that can be triggered by new events in the world, and has limited information and limited time for dealing with most of the problems. A conclusion of that argument is that the kinds of mechanisms that sometimes generate emotional states are not design defects but inevitable consequences of optimising the design, subject to all the problems and constraints of real life.

What should AI do about this?

One reaction to all this is horrified rejection and a frenzied search for some general way of using formal, mathematical methods to avoid the problems. An alternative reaction is to recognize that a major task for AI is to find ways of designing intelligent agents that reduce, even though they cannot eliminate, the bad consequences of all this scruffiness. This is compatible with trying to identify classes of sub-problems that are completely amenable to rigorous mathematical treatment.

Unfortunately, because we don't know enough about the world to prove that any particular way of minimizing the consequences is optimal, and I suspect that there is NO practically implementable strategy that will be optimal for ALL situations, it seems to me that our best hope is to see what has emerged as a result of millions of years of empirical exploration of the space of possible designs. I.e. we should study existing animals, including especially human beings, to find out how they cope reasonably well in a rich, partly knowable, partly friendly, fast changing environment.

It is unlikely that they do this by proving theorems, even theorems in non-monotonic logics.

Of course, I am not arguing that mathematics and logic are irrelevant to AI. The whole history of science has shown that on the contrary a mathematical approach is often essential for understanding a complex system, and for many aspects of AI it is clear that mathematical analyses of problems and techniques plays an essential role both in understanding the nature of the problem to be solved and in designing and evaluating representations and algorithms. My point is only that precisely defined representations and rigorous fully analysed methods have to be restricted to those sub-problems where they are applicable and useful (e.g. aspects of low level vision, design and analysis of various forms of neural computation, some aspects of motor control, complexity analysis, and so on) and that for many features of the design of intelligent systems the *working* system will have to violate canons of mathematical and logical acceptability, but for good, principled, reasons.

Conclusion

Although it may be impossible to build truly intelligent systems that never become scruffy as they struggle to understand the world and how to deal with it, at least we have a neat theory as to why it's impossible.