

Design Spaces, Niche Spaces and the “Hard” Problem

Aaron Sloman

School of Computer Science & Cognitive Science Research Centre
The University of Birmingham, B15 2TT, England

A.Sloman@cs.bham.ac.uk, <http://www.cs.bham.ac.uk/~axs>

Abstract

This is an attempt to characterise a new unifying generalisation of the practice of software engineers, AI designers, developers of evolutionary forms of computation, etc. This topic overlaps with theoretical biology, developmental psychology and perhaps some aspects of social theory (yet to be developed!). Much of theoretical computer science follows the lead of engineering intuitions and tries to formalise them. Likewise there are important emerging high level cross disciplinary ideas about processes and architectures found in nature that can be unified and formalised, extending work done in Alife and evolutionary computation. This paper attempts to provide a conceptual framework for thinking about the tasks. Within this framework we can also find a new approach to the so-called hard problem of consciousness, based on virtual machine functionalism, and find a new defence for a version of the “Strong AI” thesis.

Introduction

Robin Milner claims (Milner 1996) that much of theoretical computer science follows the lead of engineering intuitions and tries to formalise them. Since the intuitions are often very subtle and complex the process of formalisation can lag very far behind. This is just another example of the problem frequently encountered in AI: formalisation and mechanisation of powerful human capabilities can be very difficult.

This paper puts forward a related idea, namely that the processes of biological evolution have developed what can be thought of as a collection of engineering skills, methods, resources, and re-usable designs and design techniques. These too can be the subject of attempts to automate and formalise, and indeed this is one view of what is going on in artificial life, theoretical biology and the current surge of activity in connection with evolutionary computation including genetic algorithms, genetic programming, classifier systems and no doubt others not known to this author.

The notion that evolution is something like a designer is not new, and neither is the idea that the biosphere is itself a sort of organism which is struggling

to develop itself against the potentially destructive and disruptive forces of nature (and perhaps human activities). These ideas may seem far fetched and totally unworthy of scientific consideration. However, I suggest that they may provide a framework for important new multi-disciplinary scientific investigations if handled correctly.

There are a number of key ideas which need to be combined in presenting this viewpoint. One which is familiar to computer scientists and software engineers¹ is the idea of a *virtual machine*, such as the Lisp virtual machine or a Prolog virtual machine, or the virtual machine in a multi-function word processor or operating system or even the internet. These virtual machines have functional architectures; events and processes with causal powers occur in them; and they can influence and be influenced by events in other sorts of machines, including both the physical environment and the underlying physical machines in which they are implemented. The relation between a virtual machine and its underlying implementation is not unlike the philosophers’ notion of “supervenience” which is often used in discussing the relation between mind and brain e.g. in (Chalmers 1996). We propose that virtual machines can be found everywhere, in social systems (e.g. the link between poverty and crime), organisations, ecosystems, and animal’s information processing architecture.

The second key idea is a generalisation of the main theme of the presidential address by Randall Davis at AAAI96 (Davis 1998) which presented a view of AI as exploration of “design spaces” of intelligence. (See also (Sloman 1995).) This is based on the fairly familiar space of possible designs (where the notion of “design” does not presuppose any designer: a design is simply an abstract specification which can be instantiated in working systems that fit the design).

The generalisation of this idea has two features. The first is that there are *two* linked spaces inhabited by behaving systems, design space and the space of sets of requirements, which can be named “niche space”, since

¹Though some disagree with the use of this notion to extend the ontology of physics. There is no space to marshal all the arguments here.

it is possible to view the biologist's notion of a niche as close to the engineer's notion of a set of requirements and constraints. Both are abstract spaces. E.g. two insects or two plants in the same physical location can be in totally different niches, so niches are not determined by physical location. Neither are they simply in the eye of a beholder: niche-pressure can influence evolution of design traits through natural selection. Instances of niches and designs interact *causally*.

Niches can interact with one another by influencing adaptive designs satisfying those niches, e.g. as in co-evolution of organisms. Thus there are many different sorts of causal relations: within an architecture, between architectures, between architectures and niches, between niches. A niche that influences a design can be thought of as an attractor in a phase space. That's a special case insofar as phase spaces typically have a uniform topology, e.g. fixed dimensionality.

Just as designs don't presuppose a designer, so in principle requirements (niches) don't presuppose a requirer. There are different ways actual requirements can be generated: e.g. engineering goals vs biological needs and pressures.

Dynamics of niches and designs

Since niches and designs interact dynamically, we can regard these spaces as corresponding to virtual machines in the biosphere consisting of a host of control mechanisms, feedback loops, and information structures (including gene pools). All of these are ultimately implemented in, and supervenient on physics and chemistry, but real enough in their causal interactions.

This suggests a view of the biosphere as a very complex abstract dynamical system, which itself is composed of many smaller dynamical systems. Some of the sub-systems are evanescent (e.g. tornados), some enduring but changing over diverse time scales (e.g. fruit flies, oak trees, species, ecosystems). Many of the subsystems impose constraints and requirements to be met or overcome by other subsystems. Thus one component's design is part of another component's niche. Through a host of pressures, forces and more abstract causal relations, including transfer of factual information and control information, systems at various levels are constantly adjusting themselves or being adjusted or modified. We therefore have a network of designs and niches interacting concurrently at various levels of abstraction, producing a host of changes of many sorts, some continuous, some discontinuous.

Some of the changes may be highly creative, including evolution of new forms of evolution, and discovery of powerful reusable modules, and mechanisms for copying and later modifying modules to extend a design. It is conjectured below that this accounts

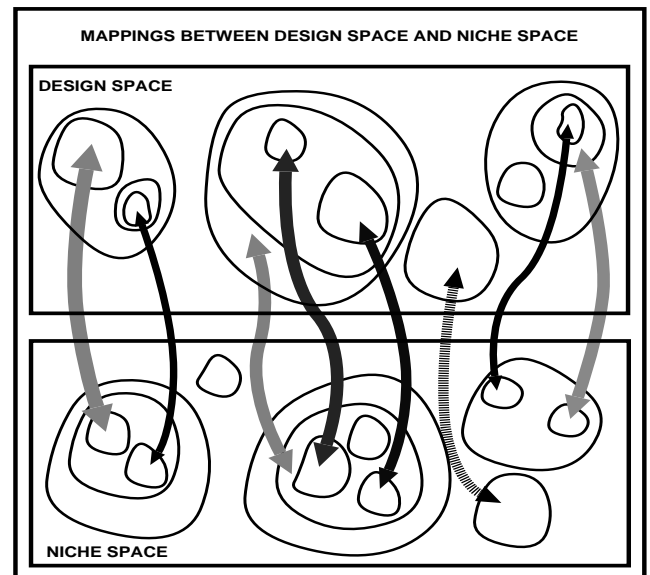


Figure 1: Design space and niche space

for the evolution of human information processing architectures.

These may seem to be outrageously wild ideas. The claim is that by exploring them we may come to see them as natural extensions of ideas already accepted by many scientists and engineers, e.g. (Cohen & Stewart 1994), defining a vast new domain for multi-disciplinary exploration.

Discontinuities and inhomogeneities

Both design space and niche space have very complex topologies, including many discontinuities, some small (e.g. adding a bit more memory to a design, adding a new step in a plan) some large (adding a new architectural layer, or a new formalism).

E.g. I suspect a major discontinuity in niche space occurs somewhere between systems that are able merely to perform certain tasks, and others which also have and can use information on how they did it, or why they did it, or why they used one method rather than another. The hybrid architectures described below might combine such capabilities. What sorts of niche pressures might lead to these design features in nature is an interesting biological question, separate from how the systems function after they have evolved.

Both are profoundly inhomogeneous spaces: the local topology varies depending on the location in the space, since the minimal changes possible at various locations in the same space can be very different in type and number. This is partly a function of complexity. Even if designs of different complexity are in the same space, there are typically more ways and more complex ways, of altering a complex design than a simple design. So they have

neighbourhoods of different structures. By contrast, in most multi-dimensional spaces considered by scientists and engineers (e.g. phase spaces), each point has the same number of dimensions, i.e. the same number and the same types of changes are possible at all points. (Equations defining permitted trajectories in a high dimensional phase space may, however, make the space inhomogeneous.)

Both design space and niche space are “layered” in that regions within them can be described at different levels of abstraction and for each such region significantly different less abstract “specialisations” exist. Some specialisations of designs are called implementations. As suggested above, the philosopher’s notion of “supervenience” and the engineer’s notion of “implementation” (or realisation) seem to be closely linked, if not identical (as argued in (Sloman 1998 in preparation)).

Composition of niches and designs

Two (or more) niches in different sub-spaces can be combined to yield a niche in a more complex space: requirements for editing programs and requirements for editing latex source files can be combined with each other and with requirements for reading and posting email and net news. Some designs for text editors (especially programmable editors like Emacs) can satisfy the composite niche, others not.

It is not in general necessary to combine two separate designs to get a new design that satisfies two separate niches. Often designs have intrinsic generality or multi-functionality: a powerful example is the architecture of a typical modern CPU, or a programming language.

In some cases it may not be clear whether we are dealing with one niche-space with enormous internal complexity or simply a collection of different niche-spaces, some of which are compositions of others. It is very likely that humans, for instance, cannot be thought of as instantiating a single design fitting a single niche, but rather multiple designs related to multiple (internal and external) niches (a view consistent with (Minsky 1987)). Both designs and niches change over the lifetime of an individual, as explained below, and may vary considerably from one culture to another.

Mappings between the spaces

Relationships between a design space and a niche space also have a very complex form: e.g. two regions of design space may be mapped to the same region of niche space, via a “satisfaction” mapping with different dimensions, and different degrees of satisfaction in those dimensions. i.e. trade-offs. The different styles of arrows in Figure 1 are intended to indicate this.

Similarly two different niches may be satisfied (to some extent) by the same design, for instance providing shelter from cold and protection from predators. When

that happens the design also satisfies a composite niche, as mentioned above.

Satisfaction thus described is more like a relation than a function.

The notion of a relation is not general enough, however, for a relation either holds or does not hold between two things. It would be more accurate to regard satisfaction as a function from a composite design and a composite niche to a partially ordered set of descriptions of the type and kind of satisfaction: “good protection from predators but mediocre winter shelter”.

Like designs and niches, these descriptions of “fitness” can vary in complexity, and their partial ordering will correspond to the notion of one design being “better” than another, which is generally a partial ordering (Sloman 1969; Logan & Sloman 1997).

By contrast, genetic algorithms and other forms of evolutionary computation normally assume the “fitness function” produces a simple numerical measure of goodness, i.e. a total ordering with a metric. We can now see that as a special case.

Intertwining spaces

Design space and niche space seem to “include” parts of each other insofar as a design may include a complex architecture where part of the architecture is a sub-architecture which “inhabits” a niche determined in part by the requirement to fit in with the rest of the architecture, and serve its needs.

Likewise a complex niche may include as one of its sub-requirements, a requirement to accommodate something with a particular architecture, e.g. provide maintenance for it. So a niche, i.e. a set of requirements can include reference to a design.

Trajectories and dynamics

There is a dynamics of niches and designs, since processes can occur involving trajectories in both spaces. E.g. besides the behaviours of a particular design instance when it does its job, there are also self-modifying behaviours which involve following a trajectory in design space, enabling new types of jobs to be performed.

A system which develops, learns or adapts changes its design. The most familiar (and dramatic) examples are biological: e.g. a fertilised egg transforming itself into an embryo and then a neonate. Other examples include self-adapting communication networks, adaptive interfaces, AI “learning” systems, and socio-economic systems.

In many animals, including humans, the information processing architecture seems to continue being transformed long after birth, and after the main physiological structures have been established: new forms of control of attention, learning, thinking, deliberating, develop after birth. Humans follow a very complex trajectory in design space throughout their

lives. A good educational system can be viewed as providing a trajectory through niche space which will induce a trajectory in design space in self-modifying brains. A culture provides a set of developmental trajectories.

In general, following a trajectory in design space also involves a trajectory in niche space: the niches for an unborn foetus, for a newborn infant, a schoolchild, a parent, a professor, etc. are all different. Moreover, an instance can instantiate more than one design, satisfying more than one niche: e.g. protector and provider, or parent and professor. Thus development of multi-functional designs can involve multiple concurrent trajectories in design space and niche space. The evolution of such systems may have facilitated production of animals able to cope with varied environments and problems without requiring all the required information to be pre-stored in genes. Compare the genetic information load of animals, like deer, which need to be able to run with the herd soon after birth and animals, like primates, which can learn to see, manipulate things, and move about, whilst under parental care. Compare genetically determined forms of representation with those developed through interaction with an environment, including peers.

Possible and impossible trajectories: i-e- and r-trajectories

Some regions of design space are not linked by possible trajectories for individual development. An acorn can transform itself into an oak tree, and by controlling its environment you can slightly modify what sort of oak tree (e.g. how big). But no matter how you try to train or coax it by modifying the environment, it will never grow into a giraffe.

Trajectories that can be followed by a self-modifying individual can be called *i-trajectories*.

If two designs that cannot be part of the same *i*-trajectory, are instantiated in a larger ecology including Darwinian reproductive mechanisms operating on collections of designs, then trajectories may be possible over *generations* of individuals that are not possible within an *individual* e.g. development of humans from much simpler organisms and genetic algorithms modifying software. Trajectories in design space that are not *i*-trajectories but are enabled by Darwinian evolutionary processes can be called *e-trajectories*.

There may be some trajectories that are impossible both for individual development and for Darwinian evolution, but can be achieved by external intervention. Plausible examples are repairing certain kinds of physically damaged system, and fixing certain kinds of bugs in software. Trajectories requiring such external "repair" can be called *r-trajectories*. It is not clear how useful these distinctions will turn out to be. From a different view point *e*- and *r*-trajectories may be *i*-trajectories for a species, or an ecology, or the biosphere.

Trajectories for virtual machines

Whether a similar distinction between *i*-trajectories and *e*-trajectories can be made for individuals inhabiting software virtual machines depends on whether the impossibility of certain trajectories depends on the individuals being implemented in physics or whether they are logical impossibilities. E.g. the acorn (1) lacks information needed by a giraffe, (2) lacks the architecture to absorb the information, no matter how the information is presented by the environment, and (3) lacks the architecture required to modify itself into an architecture that can absorb the information.

Similar limitations may be found in some software systems. A word processor which adapts itself to different users may be incapable of turning itself into a compiler. Whether an *e*-trajectory from its design could lead to a Prolog compiler depends on (a) whether there is a principled way of mapping its features and the features of compilers onto a class of structures which can be used to recreate design instances via an instantiation function, such that (b) the structures can be manipulated by processes like crossover and mutation, so as to traverse a trajectory in what might be called "gene space" which induces a trajectory in design space via the instantiation function. It's a separate question whether some sort of evaluation function or niche pressure can cause the traversal to occur – a search control problem (Poli & Logan 1996).

Virtual machine architectures

Milner's paper progresses towards an important concept in design space, namely "architecture" in the sense in which software, a sonnet, a sonata, a house, or a commercial organisation can have an architecture. A system with an architecture is "nearly decomposable" into a collection of coexisting smaller interacting systems. In particular, networks of coexisting interacting processes form architectures.

Milner's PI calculus represents systems with changing architectures, described in terms of processing nodes and the links between them. Many software architectures change at run time. E.g. Unix and other operating systems allow the creation of new processes and killing of old ones, and also creation of new links and deletion of old ones (e.g. sockets). More interesting cases would be integrated systems with a coherence not normally found in a collection of Unix processes.

In order to describe regions of design space containing architectures for integrated intelligent systems we'll need to use a collection of higher order architectural concepts, including concepts concerned with high level self-modification (e.g. the ability to remember and return to unfinished tasks, seems to be missing at first in human children).

There are concepts referring to functional roles within control systems of many types, e.g. monitoring, positive and negative feedback, adaptation, growth,

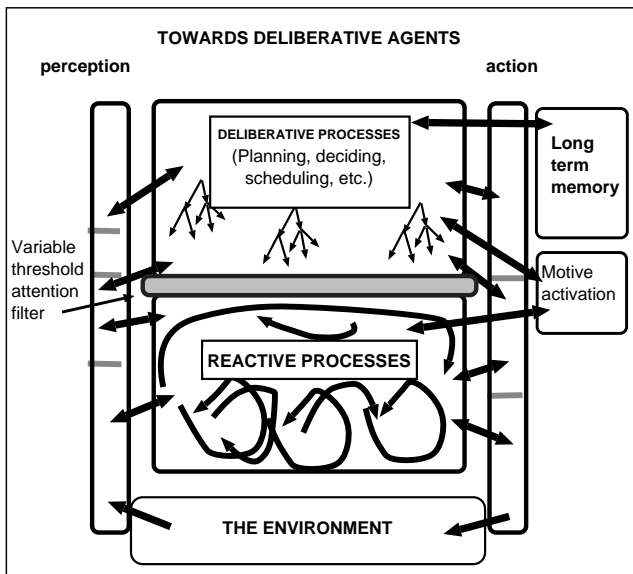


Figure 2: A hybrid reactive and deliberative architecture

record keeping, sensing the environment, repair, extension of functionality, etc. These architectural ideas need to be extended, to include mechanisms for motive generation, motive evaluation, motive selection (intention formation).

Towards human-like architectures

We have argued in (Sloman 1994a) and elsewhere that (*contra* Dennett's "intentional stance") many familiar mental concepts presuppose an information processing architecture. We conjecture that this involves several different sorts of coexisting, concurrently active, layers, including an evolutionarily old "reactive" layer involving dedicated highly parallel mechanisms each responding in a fixed way to its inputs. These may come from sensors or other internal components, and its outputs may go to motors or other internal components. Some reactive systems have a fixed architecture except insofar as weights on links change through processes like reinforcement learning. Insects appear to have purely reactive architectures implementing a large collection of evolved behaviours.

A hybrid architecture, as shown in Figure 2, could combine a reactive layer with a "deliberative" layer which includes the ability to create new temporary structures representing alternative possibilities for complex future actions, which it can then compare and evaluate, using further temporary structures describing similarities and differences. This plan-construction requires a long term memory associating actions in contexts with consequences. The deliberative system may choose one of the structures, execute it as a plan, and then discard it, or possibly modify itself permanently by saving some or all of the structure

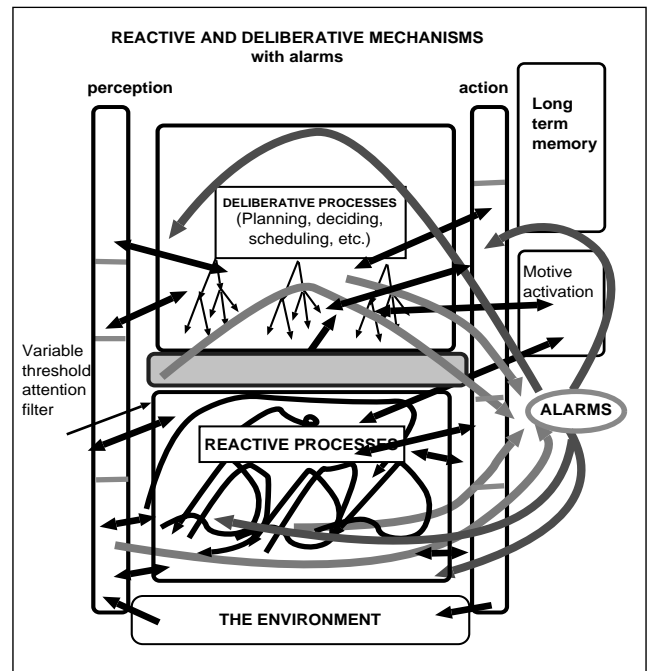


Figure 3: A hybrid architecture with global alarms

for future re-use. In humans the reactive architecture seems to be extendable in this fashion, e.g. learning car driving or language comprehension.

Evidence suggests that there is also a global "alarm" mechanism implemented in the limbic system, as indicated in Figure 3, capable of interrupting and redirecting other subsystems (e.g. freezing, fleeing, attacking, attending).

For reasons discussed elsewhere, a deliberative mechanism will (normally) be discrete, serial, and therefore relatively slow, whereas a reactive mechanism can be highly parallel and therefore very fast, and may include some continuous (analog) mechanisms, possibly using thresholds. Resource limits in deliberative mechanisms may generate a need for an attention filter of some kind, limiting the ability of reactive and alarm mechanisms to interrupt high level processing.²

²The general notion of a reactive system can be implemented in many different ways. E.g. both neural nets and parallel condition-action rules can both be seen as reactive systems. A neural net, with or without feedback, can be viewed as a highly parallel collection of condition-action rules where the conditions are some function of the inputs (e.g. a thresholded, weighted, sum) and the actions involve simultaneous broadcasts to other nodes in the network. Likewise there are many ways of implementing deliberative mechanisms, including neural nets with an appropriate architecture. As many philosophers have pointed out, at the lowest level, even deliberative architectures must have mechanisms that do not deliberate, but simply act.

Tradeoffs in design and niche space

By analysing tradeoffs we may be able to understand how niche-pressures can lead to development of combined concurrently active deliberative and reactive architectures in organisms, and also why global “alarm” mechanisms are needed.

Everything that can be done by a hybrid architecture could in principle be done by a suitably complex reactive architecture e.g. a huge, pre-compiled lookup table matching every possible history of sensory inputs with a particular combination of outputs (perhaps a probabilistically weighted selection from a set of possibilities).

However, some of the pre-requisites for such an implementation may be prohibitive: much longer evolution, with more varied evolutionary environments, to pre-program all the reactive behaviours, and far more storage to contain them, etc. For certain agents the universe may be neither old and varied enough for such development nor big enough to store all the combinations required to match a deliberative equivalent with generative power.

We conjecture that the requirements of a deliberative mechanism change the “niche” for perceptual and motor mechanisms, generating pressures for them to develop new layers of abstraction, as indicated in the figures. Likewise, development of new, higher level, abstractions in perceptual and motor systems change the niches for more central mechanisms, e.g. providing new opportunities for learning and simplified planning and decision making.

Towards “self-conscious” machines

Reflection on and retrospective evaluation of actions can often lead to future improvements. This is also true of *internal* actions. Thus besides abilities to perceive the environment and how external actions change it, there is a use also for internal self-monitoring, self-evaluation, self-modification (self-control) applied to *internal* states and processes. Depending on the tradeoffs, this may lead to special mechanisms supporting a third architectural layer, as indicated in Figure 4.

A basis for sensory “qualia” can be found in self-monitoring mechanisms which give access to intermediate sensory information structures not normally attended to. Different kinds of sensory qualia would depend on different abstraction layers within perceptual mechanisms. This provides self-knowledge in a manner which is distinct from normal perception providing knowledge about the environment. Such “meta-management” capabilities can provide other sorts of qualia related to thinking processes, deliberation, desires, etc.

The future need to support similar non-intrusive observation of part of a system by another part may require new developments in hardware and operating system features.

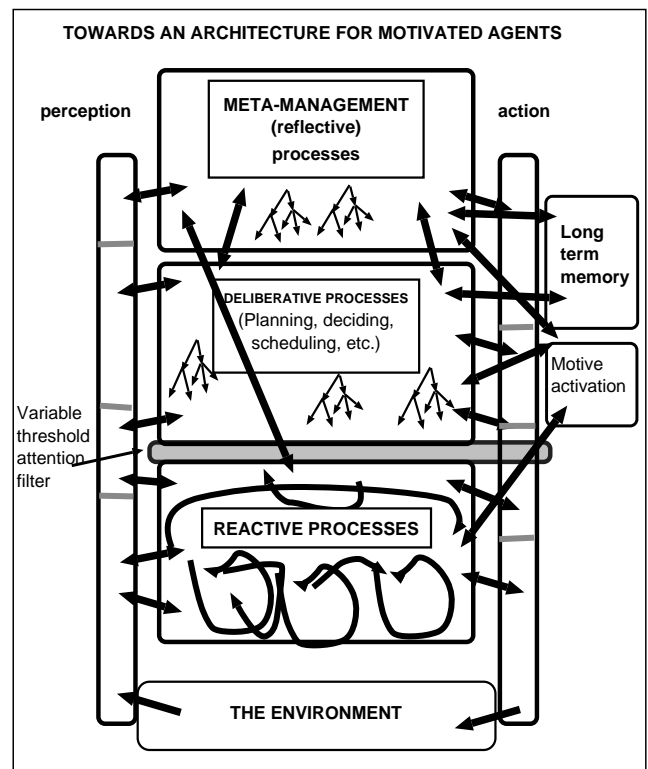


Figure 4: Adding a meta-management layer for self monitoring, self-evaluation, etc. (Alarms not shown, to save clutter.)

Robots built with this sort of functionality might begin to wonder about the nature of their own mental processes and how they are related to their physical implementation, just as human philosophers do. Some of them, not fully understanding the notion of virtual machine functionality and the varieties of forms of supervenience, might even produce spurious but convincing arguments that they have conscious processes which cannot be explained by or fully implemented in physical processes. They may wonder whether humans are actually *zombies* with all the behavioural capabilities of conscious robots, but lacking their consciousness. For more on these issues, and the supposedly “hard” problem of explaining how physical machines can produce consciousness, see (Chalmers 1996). (Some earlier work exploring these ideas can be found in (Sloman 1993; 1994a; 1994b; 1996; 1997; 1998forthcoming))

Designs for social systems

Such agents (with a combination of reactive, deliberative and self-management sub-architectures) may combine to form much larger social systems. Questions about trajectories in design space and niche space arise for social systems also.

All this seems to point to a type of theory which

unifies computer science, theoretical biology, AI, some aspects of psychology, brain science, anthropology, sociology, etc. This is still very vague, and may be too complex to reason about precisely. Yet good software engineers are beginning to develop many relevant intuitions about some of these things and it should be possible to find mathematical constructs that capture those intuitions, a process not unlike the development of theoretical concepts from programming intuitions described in Milner's paper.

Further work

We need to find ways of describing architectures and niches which correspond to the high level concepts used only intuitively at present. This will involve abstracting from domain specific details, so as to replace empirical concepts with mathematical concepts, eventually allowing mathematical analysis of relations between designs and niches.

For example, at present the requirement that a system be "user-friendly" uses an inherently empirical concept defined in terms of how humans (typical humans? all humans? humans in a particular culture? novices? experienced users?) react to a system.

In order to replace this with a non-empirical concept of user-friendliness we'll have to identify a class of agents with various perceptual, cognitive, and motivational capabilities. We can then define the various types of load a system imposes on that sort of agent, e.g. learning time required, short term memory problems, visual parsing problems, planning problems, attention control problems, etc.

Only when we have such a structural and functional characterisation of user-friendliness for a class of agents is there any hope of a deep theoretical analysis of the sort of architecture for a system that will make it user-friendly for that class of agents.

Similar ideas may enable the intuitive notion of niche, genotype and phenotype in biology to be made sufficiently precise to enable us to understand precisely the relationships between niches and designs for organisms, and perhaps give a better understanding of the dynamics and trajectories in biological evolution, including the evolution of evolvability.

Understanding the precise variety of types of functional architectures in design space and the virtual machine processes they support, will enable us to say precisely which subsets of human mental capabilities they have and which they lack. We shall also have precise new ways of thinking about human variability and the effects of brain damage, senile dementia, etc. Then instead of arguing about which animals, which machines, and which brain damaged humans have consciousness, we can find out precisely which sorts of consciousness they actually have.

References

Chalmers, D. J. 1996. *The Conscious Mind: In*

Search of a Fundamental Theory. New York, Oxford: Oxford University Press.

Cohen, J., and Stewart, I. 1994. *The collapse of chaos*. New York: Penguin Books.

Davis, R. 1998. Presidential address to aaai98. *AI Magazine*. (To appear).

Logan, B., and Sloman, A. 1997. Agent route planning in complex terrains. Technical Report CSRP-97-30, University of Birmingham, School of Computer Science.

Milner, R. 1996. Semantic ideas in computing. In Wand, I., and Milner, R., eds., *Computing Tomorrow: Future research directions in computer science*. Cambridge University Press. 246-283.

Minsky, M. L. 1987. *The Society of Mind*. London: William Heinemann Ltd.

Poli, R., and Logan, B. 1996. On the relations between search and evolutionary algorithms. Technical Report CSRP-96-07, School of Computer Science, The University of Birmingham.

Sloman, A. 1969. How to derive "better" from "is". *American Phil. Quarterly*, 6:43-52.

Sloman, A. 1993. The mind as a control system. In Hookway, C., and Peterson, D., eds., *Philosophy and the Cognitive Sciences*. Cambridge University Press. 69-110.

Sloman, A. 1994a. Explorations in design space. In *Proceedings 11th European Conference on AI*.

Sloman, A. 1994b. Semantics in an intelligent control system. *Philosophical Transactions of the Royal Society: Physical Sciences and Engineering* 349(1689):43-58.

Sloman, A. 1995. Exploring design space and niche space. In *Proceedings 5th Scandinavian Conference on AI, Trondheim*. Amsterdam: IOS Press.

Sloman, A. 1996. Towards a general theory of representations. In D.M.Peterson., ed., *Forms of representation: an interdisciplinary theme for cognitive science*. Exeter, U.K.: Intellect Books. ISBN: 1-871516-34-X.

Sloman, A. 1997. What sort of control system is able to have a personality. In Trappl, R., and Petta, P., eds., *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*. Berlin: Springer (Lecture notes in AI). 166-208. (Originally presented at Workshop on Designing personalities for synthetic actors, Vienna, June 1995).

Sloman, A. 1998(forthcoming). What sort of architecture is required for a human-like agent? In Wooldridge, M., and Rao, A., eds., *Foundations of Rational Agency*. Kluwer Academic. (Expanded version of invited talk at Cognitive Modeling Workshop, AAAI96 Portland, Oregon, August 1996).

Sloman, A. 1998(in preparation). Supervenience and implementation. Draft version available online at <http://www.cs.bham.ac.uk/~axs/misc/supervenience>.