12

Embodiment

Susan Stepney

Department of Computer Science, University of York, York, YO10 5DD, UK.
susan@cs.york.ac.uk

Summary. *Embodiment* may help to reduce the computational burden on a system, by transferring some of that burden to the complex embodying environment. Embodiment can be viewed as a property not just of situated material systems, but of any suitably complex system engaged in a complex intertwined feedback relationship with its suitably complex environment. Various features and requirements of embodiment are examined in the context of natural and of artificial immune systems. This leads to a set of suggested *design principles* for engineering embodied systems and their environments.

12.1 Embodiment : what is it?

12.1.1 Embodied in an environment

Consider a system that can sense and manipulate its environment, with its internal state depending on what it senses, and its manipulations depending on its state. Those manipulations then change the environment, and hence what is subsequently sensed, and so change the system's subsequent state and its further manipulations. This produces a complex dynamical stigmergic feedback process: the system is *embodied* in its environment.

Biological immune systems, like all biological processes, are made from physical material, they are situated in and interact with a physical environment, and they are constrained by physical laws such as conservation of matter and energy. Artificial immune systems (AIS) and other software systems, on the other hand, are informational, exist in the *virtual* world of the computer, and labour under no such constraints. Does this difference matter?

A new generation of roboticists [Brooks 1991a, Brooks 1991b], AI researchers [Maturana & Varela 1980, Varela *et al.* 1991, Simon 1996], psycholinguists [Lakoff & Johnson 1980, Lakoff 1987, Lakoff & Núñez 2000], and cognitive philosophers [Clark 1997] insist that it does. They argue that the rich dynamical interaction between a system embodied in its complex physical environment crucially gives something to the system not achieved by pure virtual or symbolic or simulated inputs alone.

Much of Brooks' robotics work has focused on the use of the environment as a resource; rather than requiring the robot to abstract data from the world, to form some impoverished model, [Brooks 1991b] exhorts the robot's designer to "*use the world as its own model*".

As an illustration of the importance of a complex environment, Herbert Simon invites us to imagine an ant walking on the beach:

> Viewed as a geometric figure, the ant's path is irregular, complex, hard to describe. But its complexity is really a complexity in the surface of the beach, not a complexity in the ant ... *The apparent complexity of its behaviour over time is largely a reflection of the complexity of the environment in which it finds itself.* [Simon 1996]

Clark also emphasises the crucial involvement of the environment:

> the deeply misguided vision of the environment as little more than the stage that sets up a certain problem. ... the environment [is] a rich and active resource—a partner in the production of adaptive behavior. [Clark 1997]

So, the environment itself is a resource, and embodiment may help to reduce the computational burden on the system itself, with some, or much, being obtained "for free" from the complex embodying environment. Of course, nothing is really for free, and the requirement for embodiment puts some interesting constraints on the design and deployment of embodied systems (later).

12.1.2 Coupled to the environment

But what precisely *is* such "embodiment"? [Kushmerick 1997, Quick & Dautenhahn 1999, Quick *et al.* 1999, Quick *et al.* 2000] note that these roboticists and AI researchers rarely bother to define the term beyond saying something like *having some (physical) body interacting with some (physical) environment*. They note that this (lack of) definition, along with the accompanying assumption of *material* situatedness, makes it particularly difficult to use the concept of embodiment when talking about Artificial Life, or virtual artefacts such as software agents.

Quick *et al.* are keen to develop a definition that makes as few assumptions as possible, in particular, one that has no requirement for the embodied system to be in some sense intelligent or cognitive or neuronal. So they turn to the concept of "*structural coupling*" [Maturana & Varela 1980], and the key idea of

> non-destructive perturbations between a system and its environment, each having an effect on the dynamical trajectory of the other, and this in turn effecting the generation of and responses to subsequent perturbations.
>
> [Quick & Dautenhahn 1999]

[Clark 1997] dubs such close structural coupling "*continuous reciprocal causation*". These ideas of dynamical trajectories, attractors, and bifurcations, which encompass both situated continuous physical processes *and* abstract discrete computations (captured in the appropriate phase or state space), are also important, implicitly or explicitly, in the writings of [Maturana & Varela 1980, Varela *et al.* 1991, Kelso 1995, Chiel & Beer 1997].

Focusing on this coupling between the system $\boldsymbol{X}$ and its environment $\boldsymbol{E}$, [Quick & Dautenhahn 1999, Quick *et al.* 1999, Quick *et al.* 2000] offer the following definition of embodiment:

> *A system $\boldsymbol{X}$ is embodied in an environment $\boldsymbol{E}$ if perturbatory channels exist between the two. That is, $\boldsymbol{X}$ is embodied in $\boldsymbol{E}$ if for every time $\boldsymbol{t}$ at which both $\boldsymbol{X}$ and $\boldsymbol{E}$ exist, some subset of $\boldsymbol{E}$'s possible states with respect to $\boldsymbol{X}$ have the capacity to perturb $\boldsymbol{X}$'s state, and some subset of $\boldsymbol{X}$'s possible states with respect to $\boldsymbol{E}$ have the capacity to perturb $\boldsymbol{E}$'s state.*

They suggest that this definition in terms of coupling can be used as a basis to *quantify* the degree of embodiment, in terms of measures of the perturbatory bandwidths and modalities, size of affected state subspaces, size of effect on state spaces, scope for variation in behaviour, structural plasticity, computational power in the interaction, and so on. They conclude that for rich embodiment one needs *both* a large perturbatory bandwidth (rich sensors and actuators to enable complex couplings with the environment) *and* large scope for variation in behaviour (complex internal dynamics for the perturbations to work on).

And most importantly for the arguments in this chapter, they note that their definition is "ontologically neutral": it is applicable to systems embodied in software (virtual) environments as well as to those embodied in material (physical) environments.

In summary, some authors argue that embodiment is essential for certain kinds of systems, and others argue that it is a property not just of situated material systems, but of any suitably complex system engaged in a complex intertwined feedback relationship with its suitably complex environment.

One feature of embodiment is that, because of the complex feedback relationship, an embodied system cannot be fully analysed in isolation: it can be analysed only in the context of the environment in which it is embodied. The fact that this environment is usually open (unbounded) has interesting consequences for such analysis.

12.1.3 Some terminology

We follow Quick *et al.*'s definitions, and so have a computational system embodied in an environment (figure 12.1).

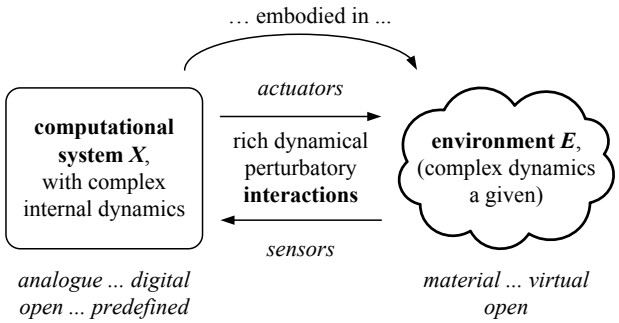

Fig. 12.1. A computational system embodied in an environment

The **computational system** itself *necessarily* runs on some form of physical hardware platform, but that hardware does not constitute the embodiment: it is the coupling with the environment that provides the embodiment. (The boundary between the computational hardware and its environment may be drawn in different places, depending on the analysis being performed. For example, in a robot, it might sometimes be useful to consider the robot body as part of the environment, and sometimes as part of the computational system.) The computational hardware platform may be analogue, digital, or a hybrid. The computational processes may be predefined (fixed code running on fixed hardware) or open (which could include dynamic binding, self-modifying code, self-reconfiguring hardware, and so on).

Following Quick *et al.*, we require that the computational system has a suitably rich complex internal dynamics. While most computational systems do indeed have a complex internal dynamics, this may not be "suitably rich", rather, it tends instead be distressingly fragile and impoverished. Engineering a *suitable* dynamics for exploiting embodiment may be non-trivial.

The **environment** in which the system is embodied may be material (for example, the physical world in which a robot is situated), or virtual (for example, the Internet). Again, a virtual environment necessarily has some underlying physical (hardware) implementation, but its behaviour may be abstractable from the details

of the particular implementation in terms of certain logical properties only. The environment is required to be *open*, hence its behaviour cannot be captured in some predefined description. A physical environment is always open: the world does what the world does, and any model we have of it is just that: a model, that abstracts away from some details. A virtual environment may also be open: for example, the capabilities and topology of the Internet cannot be captured in some predefined description, but are changing and growing in unpredictable ways on a daily basis.

The environment may be relatively passive, changing only in response to the system's actuation, or it may be following its own physical laws over time, so the effect of actuations decay or grow, or it may contain other embodied computational systems (a *social* environment), each capable of being sensed, each acting on the rest of the environment, and each altering its state based on what it senses.

The **embodiment** is provided by the coupling between the computational system and its environment, and is a rich complex feedback process. Inputs to the system from the environment are through its **sensors**, and it outputs to the environment through its **actuators**.

The effect of an actuation might be minimal, merely change the system's location or orientation in the environment, thereby changing what it senses. Such actuation is perturbing the environment in a *relativistic* sense only: we can choose to say that the actuation causes the environment to move relative to the system.

A more fully embodied effect of the actuation is to alter the environment in some explicit stigmergic way, such as by making or erasing a mark, thereby transferring a memory burden to the environment, or by building or changing a structure. Subsequent sensing, by this system or by other systems in the environment, may perceive these alterations, and thereby alter behaviours.

If the environment adapts in some way to the embodied system's change, it needs to be aware of this change, unlike in the case of mere relativistic movement. For example, a virtual environment might allocate more resources to the region that the system is currently inhabiting.

A *materially embodied system* is a computational system embodied in a physical environment; a *virtually embodied system* is one embodied in a virtual environment (that is, the terms *material* and *virtual* refer to properties of the environment, not of the computational system).

12.2 Rich dynamics

12.2.1 Physical and virtual constraints

Laws. A physical structure (here, either system X or environment E) labours under the *physical constraints* of the material from which it is constituted. These constraints include such things as: basic laws of physics, from speed of light constraints to energy density constraints; physical properties of the material, its strength and resistance; natural length scales and timescales governing the dynamics and the attractor structure of the phase space.

A virtual structure similarly labours under various *computational constraints*. The most obvious of these is *computability*. If the virtual structure in question is a Turing-equivalent machine, then the Church-Turing thesis states that it is limited to performing effectively computable functions. Some argue that this computability constraint applies to *all* physical and virtual systems; others disagree. For a good discussion of this point, see [Copeland 2002].

Questions of computability notwithstanding, the issue of *computational complexity* is crucial for virtual systems. Certain computations can be performed in principle, but in practice they take too long (longer than the age of the universe, say). This is a direct analogue of the speed of light constraint in physical systems: in principle any destination is reachable if one travels for long enough, but it may take an infeasibly long time to travel there. A class of problem whose solution time scales at most polynomially with the problem size is classed as *efficient*; one that scales exponentially with problem size is infeasible. It should be noted, however, that feasibility is a technical "worst case" measure: even if a *class* of problems is infeasible, particular *instances* of that problem may be relatively simple, and even if an exact solution is infeasible to find, a good enough approximate solution may be feasible. AIS are one bio-inspired approach to providing feasible approximate solutions to a subset of infeasible exact problems.

Even if one has efficient computation, there are further constraints when considering an embodied system. The computational system is coupled with the environment, and the environment is acting and reacting on certain timescales, constrained by the relevant laws. So it behooves the computational system to act and react on appropriately similar timescales, which puts constraints on its implementation technology. Speed matters. A Turing machine built from beer cans, or John Searle locked in his Chinese Room [Searle 1980], would not be able to constitute appropriate *embodied* intelligences with respect to our everyday environment, since they would be acting on glacially slow timescales, and could not engage in rich perturbatory interactions on our timescales.

One of the tenets of embodiment, certainly from an artificial *intelligence* point of view, is that an embodied system transfers some of its computational burden to the (much larger, much richer) environment. Thus new classes of problems may become feasible for it to solve, and others may even disappear altogether [Kushmerick 1997].

Initial conditions. Physical structures are constrained by their history and dynamics: there may be certain physical states that, although potentially physically realisable, are unreachable from the system's starting state under the physical laws of the system. [Goodwin 1994] argues that such physical constraints and processes play the major role in the evolutionary and developmental dynamics of an organism, and that genetic variation provides a relatively small modulation to this.

Virtual structures are similarly constrained by their initial conditions and the dynamics of the computation. For example, it has been found, in the context of genetic programming, that there are some potential solutions that cannot be found by an evolutionary search process, given a certain class of move function [Daida *et al.* 2003a, Daida *et al.* 2003b, Daida & Hilss 2003]. Analogous constraints are doubtless features of other bio-inspired algorithms, such as AIS, that develop from some initial population using their own move functions, although the precise forms of these constraints are yet to be uncovered.

Viability. Natural biological systems are constrained by viability: individuals must be viable at all stages of their life, and members of species must be viable at all stages of their evolutionary history. Additionally, individuals have many functions that are necessary simply for maintaining their life.

Artificial systems (virtual or physical) potentially suffer fewer such constraints. They need not be viable until their construction is complete. Also, they often need have little or no functionality beyond that needed to support their primary purpose: any resources they need merely to "survive" are usually supplied by some external agency.

As these artificial systems become more complex, however, and particularly as designers look to biology for inspiration, some of the constraints of natural systems are in turn added to artificial ones: they may be artificially evolved and grown, with some requirement for intermediate viability, and they may need to compete for resources with other artefacts.

An AIS certainly has a competitive element to it, in that the individual elements within the system may compete with other such elements; the overall system, however, is usually situated in a more computationally traditional non-competitive environment.

12.2.2 Natural physical richness

As argued above, physical and virtual systems both labour under certain constraints, and it might therefore seem that (physical) embodiment adds nothing to the equation. However, physical systems are essentially rich, whereas virtual systems tend to be *impoverished* unless specially designed for richness. It is here that the effortless richness of physical embodiment can offer new opportunities, and where analogous properties may need to be designed in to virtual embodiment.

Physical systems can exploit continual novelty. It is in this sense that they are *rich*: they can, and often do, use *any* feature of the real world to perform their task, not just the ones abstracted out for analysis in the mathematical or computational model. Physical systems can move outside the model, and evolve new representations. "*Evolution tends to produce designs that take full advantage of the available freedom*" [Beer 1995]. Computational systems embodied in a physical environment can exploit this richness.

The classic work that demonstrates such physical richness is [Thompson & Layzell 1999], who use a genetic algorithm to evolve a two-frequency discrimination algorithm running on an unclocked Field Programmable Gate Array. The resulting solution circuits perform their task, but are bizarrely inexplicable in operation. In particular, some solutions have unconnected components, yet if these apparently irrelevant components are removed, the circuit fails to operate. Also the circuits are not *portable*: they cannot be moved to other places on the chip's array, or to other chips, or run at different temperatures. The circuits appear to be exploiting *extra-logical* properties of the chips, such as capacitances between components (even ones not directly connected); these properties are not controlled by design or manufacture to be the same in all places or at all temperatures.

Even with a conventional digital microprocessor, such extra-logical properties can be important. For example, recent breakthroughs in cryptanalysis exploit extra-logical *side-channels*, for example using timing or power measurements in correlation with the computation being performed, in order to break the cryptographic systems. See, for example [Kocher 1996, Kocher *et al.* 1999, Clark *et al.* 2005b].

12.2.3 Achieving virtual richness

Physical systems do not need to develop any special mechanisms to obey the laws of nature: they just *naturally* follow such laws, and can evolve to exploit these laws. In contrast to physical environments with their open range of extra-logical properties occurring "for free", most virtual environments are severely constrained and impoverished. Typically they are *closed* systems with a pre-defined finite discrete logical representation, and cannot move out of this (or if they *do* move out, it constitutes an error).

The Internet/Web provides an interesting *open* virtual environment, in that it is constantly changing and growing. [Quick *et al.* 2000] describe *Phenomorph*, a system designed to be embodied in the open virtual environment of the Web. The system's *sensors* parse pages for keywords while its *actuators* select and navigate links. The sensory input alters the system's behaviour (in a way inspired by the locomotive behaviour of *E. coli*), which alters the actuator's choice of link to follow, which affects where the system is located in the environment.

The work is an initial attempt to explore the question: Can a purely virtual embodied system (that is, a system embodied in a virtual environment) experience the same degree of richness as a *physically* embodied system?

The answer appears to be a qualified 'yes', provided that the system and its virtual environment are designed to achieve a *suitably complex dynamics*, and that the environment is sufficiently open, allowing it to remain *far from equilibrium*.

12.2.4 Suitably complex dynamics

It is not sufficient for the system or the environment merely to have a *large* state space, or phase space: that space must also have a "suitably complex dynamics", that is, have a complex structure, neither too regular, not too random. The rich perturbatory interactions then make suitably complex changes to this structure.

The dynamics is described in terms of the trajectory that the system follows through its relevant state space. This trajectory is governed by the attractor structure of the state space. Inputs can change the system, possibly by altering the values of parameters describing the space, thereby altering its attractor structure, for example by moving attractors, or causing attractors to merge or bifurcate. [Beer 1995] provides a concise overview of the relevant dynamical systems theory.

A dynamics is suitably complex when it results in complex *emergent properties*, which may be identified with attractors or with other complex structures in the dynamics. These emergent properties are new higher level properties (patterns, agents) in space and time, and they have their own structure and dynamics, their own higher level state space, trajectories, and attractors. This higher level state space can then support the emergence of still higher level patterns, and so on. In an *open* system, where arbitrarily many levels of patterns can emerge, it is impossible to pre-define all the state space: the higher level state spaces emerge along with the patterns [Kauffman 2000], resulting in the potential for constant novelty.

Some authors argue that to achieve such richness, the state space must be continuous, rather than discrete: "*It is my belief that the versatility and robustness of animal behavior resides in the rich dynamical possibilities of continuous state spaces*" [Beer 1995]. However, research on Cellular Automata and other similarly "simple" systems shows that these discrete spaces can nevertheless have amazingly rich dynamical possibilities. See, for example, [Gardner 1970, Langton 1991, Wuensche & Lesser 1992, Wolfram 1994, Wuensche 2002].

It is an open question whether finite discrete systems can provide a sufficient richness of multiple levels of emergent structures, or whether continuous systems are qualitatively different. There has been much debate, and no doubt will be much more, on both sides of the argument.

Fractal proteins [Bentley 2004] provide one form of computational richness. The aim is to achieve the "complexity, redundancy and richness" of natural protein systems, without using (or simulating) the actual real-world mechanisms. Each fractal protein is a triplet of real numbers that encodes a small square patch of the Mandelbrot set centred at (x, y) and with size z. The rich and diverse shapes of natural proteins are mirrored in the complex and diverse shapes of these patches of the Mandelbrot set. Natural protein interactions are mirrored by intersecting fractal patches to measure their *affinity*, or how closely they match. An evolutionary or other search process searches for sets of triplets that exhibit appropriate dynamics under such intersection. The aim is for a sufficiently complex artificial chemistry, provided by the complexity of the highly non-linear Mandelbrot set.

Fractal proteins were originally designed to "provide a rich medium for evolutionary computation" of artificial gene regulatory networks (GRNs) [Bentley 2004]. Although originally designed for GRNs, fractal proteins, or related concepts, may also provide a rich medium for other bio-inspired processes, including immune ones. Antigens are a kind of protein, and [Bentley & Timmis 2004] use "fractal antigens" in the context of an artificial immune network, to represent the shape space and affinity functions.

The scheme works well in practice, but as used it may not be achieving its full potential. The search process as described [Bentley 2004] modifies the z parameter (patch size) by a random *additive* rather than *multiplicative* factor, making "deep zooming" (very small values of z) unlikely, and hence not exploiting the deep self-similar nature of the fractal. It also uses a relatively coarse-grained sampling grid on the patches to calculate their affinity. However, there is no reason in principle why a more fine-grained approach could not be used to produce arbitrarily complex chemistries.

It would be interesting to explore the effects on the generated dynamics of the precise choice of fractal, and even of (co-)evolving the underlying fractal itself.

12.2.5 Far-from-equilibrium openness

A closed dissipative system will reach equilibrium: it will converge on an attractor and stay there. If that attractor is a *strange attractor* [Lorenz 1963, Strogatz 1994], the system can exhibit complex-looking behaviour, but it will still be bounded, and in a form of "steady" state. This is why we desire the environment (at least) of our embodied system to be *open*. This openness (a constant flow of matter, energy, or information through the system) allows constant novelty, by allowing the combined system to be far from equilibrium.

A far-from-equilibrium system, rather than converging, may *self-organise* [Bak 1997] to the computational *edge of chaos* [Langton 1991]. There it can form stable structures, patterns, emergent properties, that persist; yet it is simultaneously "poised" [Kauffman 1995] in that it can readily change in response to inputs. This is what we

want from an adaptive learning system: the stable patterns form the memory, and the poised response forms the adaptation.

A physical environment is naturally open, with its noise, unpredictability, and complexity. Some virtual environments (such as the Internet) are also open. How can we provide other virtual environments with desirable open properties?

A source of *noise*, or randomness, to simulate richness is not suitable, because it has no underlying structure for the system to exploit (or, in the case of pseudo-randomness, it may have the wrong kind of structure). To get at least the flavour of the right *kind* of complexity [Crutchfield 1994], an otherwise impoverished virtual environment could be coupled to some suitable "edge of chaos" or other complex non-linear device. Depending on the nature of this device, it might still be difficult to achieve multiple levels of emergence. But in the short term, this is a potentially valuable approach to the openness problem.

12.3 Rich coupling

12.3.1 Co-evolution of sensors/actuators and processing elements

In addition to a suitably complex dynamics of the individual components (both system and environment), embodiment requires rich perturbatory channels between the systems. Similar to the argument about the state space, this does not merely mean a high bandwidth, it means a communication flow that affects the various dynamics in a suitably complex way.

If we want to engineer such a system, how can we find suitable state spaces and communication channels in the truly vast space of possibilities?

Species are not created with a given complement of sensors and actuators, filtering their interaction with their environment in fixed ways. These filters, their modalities, number and positioning, have evolved to suit the needs of the particular organism in its particular environment. Different species have different modalities (bats and dolphins "see" with sound; ants communicate by pheromones; certain fish and birds can usefully sense magnetic fields; etc), and different manipulatory appendages.

[Chiel & Beer 1997] discuss this cooperative coevolutionary history. [Percus *et al.* 1993] note that immune receptors and the antigens they sense have competitively coevolved. Kaufmann [2000] goes further, and suggests the reason why biological evolution works so well as a search technique in organism space is that organisms and evolution have themselves coevolved. Polani and co-workers [Polani *et al.* 2001, Klyubin *et al.* 2005] use a mutual information-theoretic approach to quantify informational and bandwidth requirements, given particular tasks or goals, which might be used to help evolve suitable sensors and actuators.

For artificial systems, it is possible for the environment to provide a system with inputs (or "insults") other than through its designed sensors, such as by impact, heat, etc. A sufficiently flexible system may be able to adapt to exploit these inputs, and an adaptive environment then learn to exploit them more explicitly. Similarly, an adaptive environment might learn to interpret some non-designed outputs in a useful manner: a sufficiently flexible system may be able to learn or evolve to modulate those outputs. These evolutions will simultaneously adapt the system's state space and internal dynamics.

So, for an artificial embodied system, it is essential to coevolve (or at the very least codesign) the system's computational engine along with its sensors and actuators, including the bandwidths and formats of the input/output data, in the context of the relevant environment. And ideally, for an adaptive and learning system, its sensors and actuators should be able to adapt as well.

This is contrary to the classical software engineering view that interfaces need to be clean, well-defined, controlled, and that the low-level implementation details of data representations (once a standard format has been agreed) are unimportant.

12.3.2 Co-development of system and environment

Embodiment can affect many processes, including, for example, the rate of evolution. Johan Metz [private communication, 2005] says that embodied development is a reason why mammalian morphology evolves much faster than the morphology of, say, indirectly developing insects. Mammals interact with and use cues from their environment during their development in the womb, and so bones and muscles, for example, can develop in a coordinated manner according to their use. On the other hand indirectly developing insects develop "ballistically" in the pupa, and so have no such environmental cues to exploit.

Indeed, [Riegler 2002] argues for a stronger definition of embodiment than Quick *el al.*'s. Not only must the perturbatory interactions with the environment exist, but, he claims, additionally "embodiment of a system is synonymous with competence in its environment". He goes on to claim that this competence cannot arise by engineered design, but requires "historical development in synchronization with their environment". He allows that artefacts may be embodied in virtual environments, but requires them to have developed their own goals, their own competences: designed systems whose main goals are those of the designer are merely "embedded", not embodied.

Although we do not adhere to Riegler's strong view about the source of the system goals, it is clear that the developmental process plays a key role in embodiment. Embodied systems do not spring into existence fully formed; they grow and adapt in an environment that, due to the close coupling, shapes, and is shaped by, that growth and adaptation. The same "seed", planted in two different environments, can develop into two quite different embodied mature forms. The growth itself is part

of the adaptive process of the computational system, with its complex dynamics changing as it grows.

A requirement for embodied development is rather daunting for physical systems, but may be less so for virtual ones. Recent research on virtual developmental systems might hold a key: the system is encoded as a "seed", then "grown" into its final complex adult form. See, for example, [Prusinkiewicz & Lindenmayer 1990, Kumar & Bentley 2003]. This is relevant for embodiment when the system is coupled with the environmental during its growth [Měch & Prusinkiewicz 1996].

Such an approach may help to overcome some of Riegler's arguments against designed systems. The computational system is deployed as a "seed", and grown to maturity in its specific environment, rather than designed in an explicit fully grown form; the seed itself, however, may be designed.

As a consequence, a mature embodied system cannot be simply transplanted to a different kind of environment. It must develop and learn in the relevant environment. This has been noticed in practice, for example with work on developing an AIS for fault prediction in Automatic Teller Machines (ATMs), where an AIS trained on data from an ATM in one location is ineffective when transferred to a machine in a different location [Ayara 2005]. Hence one will not be able to develop a virtual embodied system, make multiple copies of its mature form, and then deploy them in other environments, unless those environments are sufficiently similar.

12.4 Design principles for embodied systems

[Kushmerick 1997] begins a computational analysis of embodiment that could be used for the design and characterisation of virtual as well as physically embodied systems. Although his emphasis is on *intelligent* behaviour, some of the points are relevant to AIS.

One key aspect of Kushmerick's analysis is that intelligent embodied systems (animals) have a *high bandwidth* of high quality input from their environment. Vision, in particular, is a high bandwidth channel, and involves a variety of filters (such as gaze direction and attention) to control the data flow. This does not mean that all embodied systems require visual sensors: plants, for example, are embodied, but are not renowned for their sharp eyesight. What is important is the bandwidth on the relevant timescales of interaction, which is much slower for (most) plants than for (most) animals.

Another aspect is that the sensory flow is continuous: it is always present and does not have to be requested. Once an animal has focused its visual attention on a part of its environment, it does not have to "request" or "poll" for its visual input; the channel is broadcasting continuously. [Kushmerick 1997] argues that this is an important part of the interaction that allows the computational and memory burden

to be shifted to the environment (cf Brooks' "the world is its own best model"). The system lives in a sea of constantly updating data, and much of the problem of perception is what to throw away, not what to request.

[Kushmerick 1997] notes that the task that an embodied system has to perform is often more highly constrained that the general class of tasks of which it is a member, because of existing environmental constraints on the solution, or because the system itself imposes extra constraints via the coupling that reduce the number of degrees of freedom. Those constraints can then be exploited to simplify the computation needed to perform the task. (Recall that computational complexity is a worst case property of a class of problems, and individual instances may have much lower complexity.)

Finally, [Kushmerick 1997] observes that physically embodied systems are usually satisfied with approximate "good enough" solutions, rather than optimal or exact solutions, particularly when the approximate solution can be achieved with significantly reduced computational burden.

These observations, along with arguments discussed earlier, suggest some design principles for embodied systems.

1. Design the system $\boldsymbol{X}$ with sufficiently complex dynamics, that can execute this dynamics on the relevant timescale(s) of the environment $\boldsymbol{E}$.
2. Design a sufficiently high interaction bandwidth on the relevant interaction timescale(s).
3. Ensure that input from the environment is constantly available and up to date.
4. Ensure that the system perturbs the environment, rather than being merely a passive observer.
5. Ensure that the environment has sufficiently complex dynamics.
6. Allow the system to exploit structure and constraints in the environment in order to simplify its tasks.
7. Apply embodied systems only in "softer" problem domains where approximate solutions are appropriate and acceptable.
8. Co-design the system and its interface (sensor and actuator numbers, positions, data formats, etc).
9. Design the system to develop, "grow", in the relevant environment.

These principles apply both to *material* and to *virtual* embodied systems, but the emphases and difficulties are different in each case. For example, complex dynamics may be easier to achieve for a physical than a virtual environment, whereas the co-design of sensors and actuators may be easier to achieve for virtual than for physical interactions.

Of course, *how* to design and ensure many of these things are still open research problems. Certain aspects of the "design" may need to be accomplished by some evolutionary search process. However, one point to note is that some of these design principles refer to the *environment*, not just to the computational system. This is

not unexpected: we noted earlier that it it impossible to analyse an embodied system in isolation, and so it is impossible to design it in isolation. (In chapter 16 of this book, Hone and Van Den Burg analyse an isolated AIS. This does not contradict the claim here, since their AIS is not embodied: it does not have rich perturbatory interactions with its environment.)

12.5 Embodiment in the natural immune system

12.5.1 Instantiating the model of embodiment

Let us now consider the natural immune system as an embodied computational system. (The following discussion is necessarily a gross simplification of the actual biological processes, omitting many constituent agents and processes, but is sufficient to illuminate the key concepts.) We instantiate the generic embodied system of figure 12.1 with immune system concepts (figure 12.2).

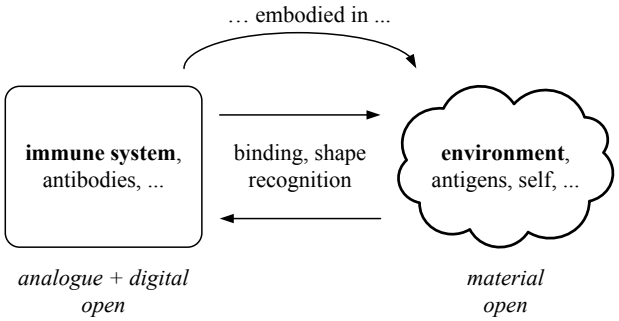

Fig. 12.2. A computational immune system embodied in an environment

The immune system contains, among other things, a population of *antibodies*, which are particular kinds of proteins. The system is hybrid: partially digital (proteins are strings of amino acids drawn from a small alphabet of possibilities, partly coded for in the DNA) and partially analogue (the proteins fold into complex three-dimensional shapes whose physical properties determine their behaviour). The system dynamics includes the modification of the population by the production, modification, and destruction of different kinds of antibodies, in response to events sensed in the environment. See, for example, [Nowak & May 2000].

The environment is the open material environment of the body, taken here as containing the antigens to be handled, and self proteins that should not be attacked. The environment is modified by the immune system, and some of its components

evolve to resist this modification, some to aid it. It is open in that there is a flow of matter through it.

The interaction between the system and the environment occurs by antibodies binding to antigens. This binding depends on shape recognition (hydrogen bonds form between the molecules; if the shapes are sufficiently complementary, enough hydrogen bonds can be created to form a strong enough aggregate bond to stick the molecules together), and other environmental factors (for example, water molecules can plug gaps between the shapes and form additional hydrogen bonds). The stronger the bond, the higher the *affinity* the antibody has for the antigen.

12.5.2 Danger Theory as embodied sensing

There may be hierarchies of embodiment. Here we have described the immune system as being comprised of its cells, with the host organism and pathogens acting as its environment. The host organism itself is also a system, embodied in its own environment. We could instead consider the immune cells to be embodied in their host, with the pathogens acting as an external environment, much in the way a robot controller is considered to be embodied in the physical robot host, in the environment of the world. However, in our case, such a choice would beg the question of precisely what comprises the host organism, and what comprises the pathogenic environment.

There are several theories, all more or less contentious, of how the natural immune system determines what of its environment is host organism, and what are the pathogens. These range from externally directed self-non-self learning, to fully autopoietic ideas of self-assertion (summarised in [Bersini 2002]).

Danger Theory [Matzinger 1994a, Matzinger 2002] sits towards the middle of this spectrum, and says that the immune system reacts to danger, or rather, to damage, by reacting to certain chemicals given off by distressed cells. When it senses such a "danger signal", it reacts by associating "nearby" cells with the danger, and so they are classified as the cause of the problem, in a form of "guilt by association". Danger Theory helps to explain why an adjuvant (essentially, a poison) is a necessary part of some vaccines: it causes the required damage.

In Danger Theory, the immune system has cells with sensors that detect danger signal chemicals: specific chemicals given out by damaged or killed host organism cells. It also has sensors to detect nearby cells, and a system dynamics that correlates the inputs from these sensors. Additionally, it has actuators (further cells) that then respond to (kill) these associated cells, whether or not they subsequently occur in proximity to any danger signal.

Under this theory, the immune system has evolved its internal computational dynamics and its sensors to respond to danger signals. One can also postulate that the host organism part of the environment may have coevolved to enhance these signals, and that the pathogens have coevolved to evade the associated recognition, provid-

ing the environment with its own complex dynamics. Thus we see the evolution of a rich embodied computational system.

Ideas from Danger Theory are now being incorporated in AIS models. See, for example, [Bentley *et al.* 2005, Greensmith *et al.* 2005]

12.5.3 Shape space in the natural immune system

As noted earlier, the interaction between the immune system and its environment is in terms of shape recognition.

[Perelson & Oster 1979] introduce the concept of an abstract **shape space**, S, in which the shape of an object is represented by a single point. They take S to be an N-dimensional Euclidean vector space, $S = \Re^N$. They use the N shape parameters as a model of the relevant features of an antibody combining region, and represent these values by a point $\mathbf{Ab}$ in S. They do not identify these parameters further, beyond suggesting that they could be physical properties such as size, charge, and dipole moment. [Lapedes & Farber 2001] use multidimensional scaling techniques to derive further geometrical properties of shape space from influenza data.

[Perelson & Oster 1979] then capture how well an antibody combining region $\mathbf{Ab}$ and antigen $\mathbf{Ag}$ fit together in terms of the *distance* between the shape of the antibody and the complementary shape of the antigen, $\|\mathbf{Ab}, \overline{\mathbf{Ag}}\|$, measured using an "appropriate" metric on S. A small distance represents a good fit, and hence a high *affinity*. They explicitly decline to define this metric, its being "a complicated chemical problem", and continue their analysis in terms of the well-known Euclidean metric. (See appendix A for some other possibilities.)

What is the distribution of antibodies and antigens in this shape space? Perelson and Oster note that it is almost certainly not *random*, since the components will have been subject to negative selection (in order not to recognise *self*) and evolutionary selection respectively. It is presumably also affected by the physical constraints of embodiment: certain combinations of parameters may be physically impossible, or at least highly unlikely, to be realised.

Since nothing is known of the actual distribution, analyses tend to proceed on the minimal assumption of random distributions within some finite volume $V \subset S$ (or are parameterised by the actual, but unknown, distribution). Perelson and Oster analyse the size of the antibody repertoire needed to cover this volume, in terms of a recognition specificity $\hat{\varepsilon}$ and of the dimensionality N. Their analysis shows that higher values of N need significantly larger antibody repertoires to cover a given volume of shape space, for a given specificity. They argue that this is why recognition uses only a small portion of each antigen, to limit the complexity, and hence to limit N and the required size of the repertoire. For typical animals, they estimate that $N = 5 - 10$.

Using different techniques, [Lapedes & Farber 2001] recover $N = 5$ from various influenza data. [Smith *et al.* 1997] find that $N = 5 - 8$ is consistent with their data; using a Hamming metric (see appendix A) instead of a Euclidean metric, they find $N = 20 - 25$, for an alphabet of size $3 - 4$, is consistent with their data.

[Perelson & Oster 1979] define their recognition specificity $\hat{\varepsilon}$ as the radius of an N-sphere centred on **Ab** in S, scaled by the radius R of the full N-sphere of volume V, making it a non-dimensional quantity with $0 \leq \hat{\varepsilon} \leq 1$. The number of such spheres needed to fill V goes like $1/\hat{\varepsilon}^N$. Keeping $\hat{\varepsilon}$ fixed whilst varying N corresponds to a keeping a fixed specificity along each individual dimension whilst changing the number of dimensions. This form of scaling is (most of) the reason for the dependence of repertoire size on N: if one fixes specificity along each dimension whilst increasing N, the number of antibodies in the repertoire needed to cover the space increases.

One can look at this argument from another perspective. Consider a *fixed repertoire size* $1/\hat{\varepsilon}^N$; then as N increases, $\hat{\varepsilon}$ *also* increases. For example, for a repertoire size of 64, then $N = 1$ has $\hat{\varepsilon} = 1/64$; $N = 2$ has $\hat{\varepsilon} = 1/8$; $N = 3$ has $\hat{\varepsilon} = 1/4$; and $N = 6$ has $\hat{\varepsilon} = 1/2$. So, for a fixed repertoire size, one can trade off specificity against dimension: a larger N requires less precise discriminations along each of its individual dimensions, which might be easier to realise.

Shape space is an essentially physically embodied notion, as further demonstrated by the following quotations from [Perelson & Oster 1979] (our italics):

> multisite recognition is a more reliable method of distinguishing between *molecules* than single site recognition. This may have been an important evolutionary consideration in the selection of *weak non-covalent interactions* as the basis of *antigen–antibody bonds*.
>
> ... a large repertoire of *antibody molecules* with different *three-dimensional binding sites* specific for the *different chemical groups* ... found on *antigen molecules*.
>
> because of *physical restrictions* on the manner in which *molecules fold*, smaller regions can take on fewer shapes

Shape space as originally envisaged is a metric space, where the only property of interest is the distance between pairs of points in the space. Spaces with more structure, with a value at each point (for example a scalar value that might represent a fitness, or a vector value that might represent a force), are also of potential interest for modelling the interaction between a system and its environment. Entities can then have a dynamics, can move around in these spaces, affected by the values at various points. The values in turn can be affected by how things move, and thus provide even richer dynamics [Saunders 1993]. In Chapter 4, Lee and Perelson discuss further models of affinity, and their dynamics.

Complementarity revisited. Care needs to be taken with the various forms of complementarity used in different shape space arguments. The idea is that com-

plementary shapes match well. However, the complementarity may be included at different points in the argument, resulting in different interpretations of the metric.

The original shape space arguments [Perelson & Oster 1979] consider a specificity "ball" around an antibody, with this ball containing the antigens recognised. The antigens are located in the shape space according to the complement of their shape. The Euclidean metric measures how well antigen and antibody match, where *small* values correspond to good matches, that is, to high affinity. So in this case, high affinity corresponds to small values of $\|\mathbf{Ab}, \overline{\mathbf{Ag}}\|_E$. [De Boer *et al.* 1992] take a slightly different interpretation: shapes have good matches in shape space when their coordinates are "*equal and opposite*" (thus putting an explicit interpretation on complementarity), so high affinity corresponds to small values of $\|\mathbf{Ab}, -\mathbf{Ag}\|_E$.

The "lock and key" metaphor [Percus *et al.* 1993] has a more direct measure of complementary shapes matching well. Here r-contiguous bits (which measures complementary contiguous subsequences) is used, and *high* values correspond to good matches. So high affinity corresponds to large values of $\|\mathbf{Ab}, \mathbf{Ag}\|_C$.

In practice, it is common for *artificial* immune system researchers to neglect complementarity when using real-valued artificial shape spaces, and instead measure direct matching, and use a large (or sometimes, small) value of $\|\mathbf{Ab}, \mathbf{Ag}\|_E$ to represent a high affinity.

Discussions of the role of complementarity and metrics in the context of artificial immune systems can be found in [Garrett 2003, Hart & Ross 2005]. It is clear that great care should be taken in choosing, and precisely documenting, what measure of matching is being used.

12.6 Embodied artificial immune systems

Engineered embodied systems must have their own suitably complex internal dynamics and complex perturbatory interactions with a suitably complex open environment. This is a potentially enormous class of systems, and so we seek inspiration from existing natural systems to help us to constrain our design space.

As discussed in Chapter 3, Artificial immune systems (AIS) are computational systems that take their inspiration from various theories of the natural immune system. However, research has tended to concentrate more on the complex internal dynamics than on the complex perturbatory interactions, and the systems are often situated in a closed and impoverished virtual environment: AIS tend not to be *embodied*. Let us look at the design principles suggested earlier, to see how they might be used to achieve (materially or virtually) embodied AIS.

Timescales. Physically embodied systems usually interact on "human" timescales (seconds, hours, days), whereas virtual systems may have to react much faster.

There may be more than one relevant timescale, for example, for immediate reaction to a novel situation, and for slower adaptation/learning of that novelty. One of the reasons that AIS have not managed to act as fully effective "immune system" defences against computer viruses may be that, although the analogy holds between the two sets of terminologies, it does not hold between timescales. A computer virus can destroy an entire network on the *same* timescale that the defences react.

Bandwidth. A physical environment naturally offers high bandwidth, so the AIS merely needs to be supplied with sensors and actuators to access the relevant portion of it. A virtual environment may need to be adapted to provide a richer source of data: not simply, say, packet headers, process ids, or return codes, but more contentful information. This may pose a problem on, say, a network where all traffic is encrypted. It is worth considering if "side channel" information (such as timing, power consumption, etc [Kocher 1996, Kocher *et al.* 1999, Clark *et al.* 2005b]) are also available, as these provide rich extra-logical resources.

Constantly available up-to-date input. The AIS should be able to directly sense the relevant data streams, rather than explicitly request data. The requirement to be up to date might imply that virtual data streams have limited temporal and spatial extent, so that they "decay" before they have become outdated.

Actuation. An embodied system perturbs its environment: it takes actions based on what it senses, and these actions in turn change the environment. The AIS must interact, and be able to change aspects of the environment (for example, by injecting packets into relevant data streams). Thus purely passive recognisers or monitors (the major application areas of AIS to date) cannot be considered to be embodied.

Sufficiently complex environmental dynamics. This should be a "given" for truly physical embodiment. However, in some cases the physical world may instead be severely impoverished, by design. For example, many robot control experiments take place in sterile featureless mazes that offer few environmental cues. These are not appropriate environments for embodiment.

A virtual environment may have to be designed explicitly to have sufficiently complex dynamics; this may be simulated by artificially hooking up the environment to an edge of chaos generator. "Sufficient" complexity allows the actuations of the system to affect the dynamics of the environment in complex ways.

Environmental structure and constraints. Exploiting these requires an understanding of the constraints (for example, restricted network topology resulting in restricted navigational opportunities) and their effects, and how they might change over time.

Approximate solutions. In most physically embodied cases, approximate solutions are acceptable, because the system is interfacing with the continuous, analogue real world, and there is always limited precision. The acceptable degree of approximation still needs to be ascertained. In the virtual world, crisp digital problems are not appropriate: this excludes many classical computer domains, from word pro-

cessing to payroll systems. Even for softer domains such as pattern recognition, it requires consideration of the acceptable false positive and false negative rates.

Co-design the system with its sensors and actuators. Natural immune systems interact with the environment by shape recognition and binding, modelled by shape space. The analogue for AIS is an artificial shape space (sensing affinities), and artificial binding (actuation).

Relatively simple shape space arguments show how the choice of geometrical parameters such as representation dimensionality, alphabet size, and specificity, determine the antibody repertoire size needed to cover shape space. This is a crucial design consideration for artificial immune systems. Evolution appears to have settled on a value of $N \sim 5$ for real immune system shape space. Design choices include the actual dimensionality N, what each axis of dimension represents, the metric used to measure affinity distance, and the specificity $\hat{\varepsilon}$.

The data representation also matters. For example, interpreting a b-bit string as representing a low dimensionality, large alphabet system, rather than as a b-dimensional binary hypercube, results in smoother measures [Smith *et al.* 1997], which may be beneficial to the behaviour of the system. It may be that the use of complementarity and the choice of metric (or non-metric distance measure, see appendix) have subtle effects on the underlying dynamics of the system [Hart & Ross 2005], and so should be designed into artificial systems with care.

Grow the system in the relevant environment. Essentially this affects the initial conditions of the system. A system should start as small as possible (maybe from an embryonic or an infant state), and develop in the context of its environment, learning and adapting as it goes. It should not merely accrete: the dynamics of its growth should be complex, and affected by the environmental inputs. It should be a continually developing system. This implies that the system should undergo continual online learning, and not be a system that has an initial learning phase, then a frozen deployment phase.

12.7 Conclusions

Embodiment offers substantial advantages in allowing a computational system to offload much of its computational burden to the surrounding rich environment. This advantage comes at a price, however, as the design of such a system is non-trivial.

Despite embodiment seeming to require a physical body, many of the advantages can also be achieved in systems embodied in a virtual environment, provided that the design of both system and environment supports the rich dynamics and interactions necessary.

Current artificial immune systems tend not to have the properties necessary for embodiment. Several design principles for embodied systems have been abstracted, and their application to potential embodied artificial immune systems explored. The next step is to attempt to build some fully embodied artificial immune systems according to these principles.

A Metrics

A *metric* is the mathematical abstraction of the intuitive concept of *distance*. It is a function that maps pairs of points x, y in a space S to a real number, the "distance" between them: $\|x, y\| \in \Re$. To qualify as a *metric*, the function must obey the following properties.

1. distances are not negative: $\forall\, x, y : S \bullet \|x, y\| \geq 0$
2. the distance from a point to itself is zero; the distance between different points is not zero: $\forall\, x, y : S \bullet \|x, y\| = 0 \Leftrightarrow x = y$
3. distance is *symmetric* (the distance from x to y is the same as the distance from y to x): $\forall\, x, y : S \bullet \|x, y\| = \|y, x\|$
4. distance obeys the *triangle inequality* (detours are further than direct routes): $\forall\, x, y, z : S \bullet \|x, y\| + \|y, z\| \geq \|x, z\|$

Give a prospective metric, usually the first three properties are immediate from the definition and only the triangle inequality property needs further demonstration.

There are many metrics; we consider here those most often used in the AIS literature. In what follows, we take the points in space to be vectors $\mathbf{x}$ in the N-dimensional shape space of an antibody **Ab** or antigen **Ag**. A vector $\mathbf{x}$ can be written in terms of its components x_i, with $\mathbf{x} = \sum_{i=1}^{N} \hat{\mathbf{e}}_i x_i$, where the $\hat{\mathbf{e}}_i$ are a basis set of orthonormal vectors.

The **Euclidean metric**, favoured by mathematicians because of its nice analytical properties, is

$$\|\mathbf{x}, \mathbf{y}\|_E = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

For example, $\|3\hat{\mathbf{e}}_1, 4\hat{\mathbf{e}}_2\|_E = \sqrt{(3-0)^2 + (0-4)^2} = 5$

The **Manhattan metric**, favoured by computer scientists because it is simple and cheap to compute, is

$$\|\mathbf{x}, \mathbf{y}\|_M = \sum_{i=1}^{N} |x_i - y_i|$$

For example, $\|3\hat{\mathbf{e}}_1, 4\hat{\mathbf{e}}_2\|_M = |3-0| + |0-4| = 7$

The **Hamming metric** is even simpler. Let the components of $\mathbf{x}$ be drawn from some finite alphabet Σ (so $\mathbf{x} \in \Sigma^N$ rather than $\mathbf{x} \in \Re^N$). Then

$$\|\mathbf{x}, \mathbf{y}\|_H = \sum_{i=1}^{N} \delta(x_i, y_i)$$

where $\delta(x, y) =$ if $x = y$ then 0 else 1

So, for example, $\|120001, 020020\|_H = 1+0+0+0+1+1 = 3$. For binary alphabets (that is, Boolean vectors), the Hamming metric is the *total number of complementary bits*, and is the same value as the Manhattan metric over the N-dimensional hypercube.

The ***r*-contiguous bits** distance [Percus *et al.* 1993] is also applicable to Boolean vectors. It measures the *longest contiguous subsequence of complementary bits*. The definition, expressed in the Z mathematical language [Spivey 1992, Valentine *et al.* 2004], is as follows. Let s be the sequence of vector components of $\mathbf{x} - \mathbf{y}$, that is, $s = \langle |x_1 - y_1|, \ldots, |x_N - y_N| \rangle$. So, for complementary bits,

$$\|\mathbf{x}, \mathbf{y}\|_C = \max \{ t : \mathrm{seq}\{1\} \mid t \text{ infix } s \bullet \# t \}$$

Consider the distance $\|110001, 010010\|_C$. The sequence $s = \langle 1, 0, 0, 0, 1, 1 \rangle$. So $\|110001, 010010\|_C = \max \{ \# \langle 1 \rangle, \# \langle\, \rangle, \# \langle 1, 1 \rangle \} = \max \{1, 0, 2\} = 2$.

[Percus *et al.* 1993] motivate this choice of distance in terms of the immunological *lock and key* metaphor, that the 1s and 0s of each bit string are modelling the relevant ups and downs of the key teeth and complementary lock shape, and that a contiguous run is needed to generate sufficient affinity. They also allow the shape space vectors to have non-Boolean-valued components (they find evidence for a trinary alphabet of values in natural immune systems, corresponding to positively, negatively, and neutrally charged regions), with matching being all (for complementary values) or nothing.

Formalising this, each vector component is drawn from some alphabet Σ. Each element $\sigma \in \Sigma$ has a complement $\bar{\sigma} \in \Sigma$. An element may be self complementary, $\bar{\tau} = \tau$. Complementarity is a symmetric relation: $\bar{\sigma} = \tau \Leftrightarrow \bar{\tau} = \sigma$. Now define the sequence of components as $s = \langle \delta(x_1, y_1), \ldots, \delta(x_N, y_N) \rangle$ where $\delta(x, y) =$ if $y = \bar{x}$ then 1 else 0. Then r-contiguous bits is as defined earlier, using

this s. For example, consider the alphabet $\Sigma = \{+, -, 0\}$ with complementarity relation $\bar{+} = -, \bar{0} = 0$. Now consider $\|-+000+, +++-0-\|_C$, which has the sequence $s = \langle 1, 0, 0, 0, 1, 1\rangle$, and so $\|-+000+, +++-0-\|_C = 2$.

It should be noted that r-contiguous bits does *not* form a metric (the notation we use above notwithstanding). To see this, consider the binary vectors $\mathbf{x} = 0000$; $\mathbf{y} = 1010$; $\mathbf{z} = 1111$. If we consider the usual complementarity relation, $\bar{0} = 1$, we have that $\|\mathbf{x}, \mathbf{y}\|_C + \|\mathbf{y}, \mathbf{z}\|_C = 1+1 < 4 = \|\mathbf{x}, \mathbf{z}\|_C$, which does not obey the triangle inequality. If we take the self-complementarity relation, $\bar{0} = 0, \bar{1} = 1$, then we have $\|\mathbf{x}, \mathbf{x}\|_C = 4 \neq 0$, which violates another of the metric conditions.