Evolution of Self-Definition

Catriona M. Kennedy
Artificial Intelligence Institute
Dresden University of Technology
D-01062 Dresden, Germany

ABSTRACT

When considering an architecture for an artificial immune system, it is generally agreed that discrimination between self and non-self is required. With current immune system models, the definition of "self" is usually concerned with patterns associated with normal usage. However, this has the disadvantage that the discrimination process itself may be disabled by a virus and there is no way to detect this because the algorithms controlling the pattern recognition are not included in the self-definition. To avoid an infinite regress of increasingly higher levels of reflection, we propose a model of mutual reflection based on a multi-agent network where each agent monitors and protects a subset of other agents and is itself monitored and protected by them. The whole network is then the self-definition. The paper presents a conceptual framework for the evolution of algorithms to enable agents in the network to become mutually protective. If there is no critical dependence on a global management component, this property of symbiosis can lead to a more robust form of distributed self-nonself distinction.

1. INTRODUCTION

It is generally agreed that artificial immune systems nust have the capability to discriminate between self and non-self. However, a much more difficult question is how "self" should actually be defined: i.e. what patterns, processes etc. should constitute it? The difficulty of this problem has been pointed by Ishida [6]. With current artificial immune system models (e.g. Dasgupta and Forrest [2]), the self-definition is based on patterns of normal usage. These patterns may, for example, be represented as strings of commands. Anything which does not match those patterns represents non-self and must be rejected.

There are two problems with this scheme. First, an anomaly may disrupt the actual operation of the immune system itself, meaning that the capability to distinguish between self and non-self *may itself be disabled*. Secondly, there is the possibility of "allergic" reaction (i.e. harmless anomalies are rejected).

To solve the first problem, it is necessary to include the controlling algorithms (those underlying the distinction process) within the self-definition. However, the "obvious" solutions to this problem leads to an infinite regress. Assuming that we base the architecture on patterns of activity, the self-definition would have to include all patterns of access to data objects including the activity pattern of the algorithm implementing the pattern recognition capability (which may be called R_0). To handle the situation where R_0 is affected by a virus and its own recognition capability is impaired, a reflective meta-level R_1 is required to detect anomalies in the activity patterns of R_0. But R_1 produces additional patterns which must also be monitored (which requires yet another meta-level R_2 and so on). It follows that there is always some part of the system which cannot be included within the self-definition and therefore remains vulnerable to attack.

This weakness has been called the "blind spot" or "reflective residue" by some in the historical cybernetics community e.g. Kaehr [3]. Possible applications of their concepts to autonomous agents and the formal description of living systems are discussed in von Goldammer et. al. [13], [14].

Although natural immune systems are not invulnerable in this way, there are convincing arguments that living systems have mechanisms to compensate for this problem or possibly eliminate it altogether. This is the view of autopoiesis theory (Maturana [10] and Varela, [12]), which emphasises the circular and non-hierarchical nature of living systems. Our model is based on some of these principles.

The main argument in this paper is that the reflective residue problem can be compensated for in artificial systems if the self-definition is something that *evolves* so that it becomes increasingly "accurate", i.e. it should represent as far as possible all processes within the system to be protected, including the immune system itself. We will also show that the mechanisms in this evolutionary process will also help solve the second problem (allergic reaction).

Our approach is similar to existing immune system models in that it also distinguishes between a maturation phase (in the sense of a "negative selection" algorithm) and an operational phase where external anomalies can occur. Apart from this, it is not based very closely on biological immune systems. One major difference is that instead of simply establishing self-tolerance, we try to evolve forms of active cooperation (symbiosis) between agents where possible, and for the whole network this takes the form of distributed self-preservation. Evolution of symbiosis is already a research topic in artificial evolution (e.g. Bull and Fogarty [1]).

Self-Definition on Two Levels

Since we are discussing evolution of self-definition, some aspects of self should be continually changing. However, there are also aspects that must be preserved against attack. For more conceptual clarity therefore, it is necessary to define self on two levels as follows:

a) *Functional* level: the capability of the system to meet a set of requirements or goals; this must be *preserved* by the immune system.

b) *Structural* level: implementations of the capability to meet the requirements; this may be changed by the immune system as a process of adaptation to anomalies. It consists of algorithms and particular representations of requirements.

Mutual Reflection

Instead of a single agent with self-reflection as in many AI systems (see e.g. Maes and Nardi [5]), we propose a network of "mutually reflective" agents, where each agent monitors and protects a subset of other agents and is itself monitored and protected by them. For example, if two agents A_1 and A_2 protect each other, the "self" from the point of view of A_1 is A_2 and vice versa. The ideal solution (no reflective residue at all) does not then involve an infinite regress because each agent is the other's meta-component. There remain only the practical limitations of current technology which may be overcome. For example, there should be no critical reliance on global management of any kind (since it would be a reflective residue).

Mutual Modification

During the operational phase (once symbiosis has developed), an agent A_i may detect that the performance of agents under its "protection" is deteriorating due to external anomalies. Then it should be possible for it to *change the implementation* of the agent(s) being affected, i.e. it must select alternative mechanisms which implement the same functionality (or learn them if they are not already available). This requires that the mechanisms implementing the functional self-definition should be allowed to change and evolve without artificially imposed restrictions. Otherwise they would become vulnerable because they are not allowed to be modified to escape attack.

2. FUNCTIONAL SELF-DEFINITION

We now consider in detail how to define (or evolve) a functional self-definition. If we return to the principle of mutual reflection above, and we assume there is a task T to be achieved by the system, the simplest possible network would look as follows:

A_1: "Meta"-agent: checks if the operation carried out by A_2 satisfies its requirements (i.e. does it continue to fulfil task T?).

A_2: "Object"-agent: carries out T **and checks if the anomaly-detection program A_1 satisfies its requirements** (effectively looks for anomalies in the anomaly-detection process).

This is the above-mentioned "meta-" and "object-" architecture with the addition of the second function of A_2 in bold face, which makes the object-component into a meta-meta-component.

However, this architecture has the problem that T is allocated solely to A_2 and if this agent fails, then the whole functionality could be lost suddenly (until A_1 can find a way of recovering it which may not always be possible). Instead, the functional self-definition should be distributed evenly among all agents over several perspectives. There are two approaches to this.

Top-Down Specification: To ensure the potential for symbiosis, the simplest solution is to specify one set of requirements (without conflicts), divide it up into subsets and allocate each subset to a seperate development team. In this case the functional self-definition is predefined by a single agency, and its evolution is restricted to refinements and modifications later in the software life-cycle.

Bottom-up Evolution: It is clearly better if the requirements are oriented around what actual users consider *valuable*. The requirements of security and efficiency can be encoded differently with varying degrees of emphasis, priorities etc. according to perceptions and interests of individual users. The functionality of the whole system may then be a "democratic" representation of user interests. If the system is complex enough, there will almost certainly be some clusters of requirements from which symbiosis can develop. Conflict management strategies for this kind of multi-perpective situation already exist in the literature, e.g. [11] and [8].

Other Applications

In addition to the obvious requirements of computer security, users normally have more specific wishes regarding content of information on the web and being able to work efficiently. An interface agent can build a model of the wishes and activities of an individual user (or a group of users) directly. A "destructive" form of outside interference would be anything which prevents the software from satisfying these user requirements (e.g. junk mail).

3. STRUCTURAL SELF-DEFINITION

The *structural* self-definition can be defined as a set of algorithms which mutually support each other in satisfying their respective requirements. We now consider how to evolve this mutual support during the maturation phase. (Requirements are assumed to remain constant during this phase).

Terminology

We use Koza's Genetic Programming (GP) [9] as an evolutionary technique. A program in a population undergoing evolution will be called a gp (lower case).

Since we have defined "self" from the functional point of view as the capability to satisfy a set of requirements, self-tolerance can be defined as "conflict avoidance". This means that in a group of agents $A_1, A_2, ..., A_n$, the actions of each agent does not decrease the effectiveness of the remaining agents in the fulfilling of their respective tasks, i.e. they do not interfere with each other destructively. (In the initial stages of evolution this will be the norm however).

Symbiosis means that the actions of each agent increases the effectiveness of the remaining agents in the fulfilling of their respective tasks, i.e. the agents interfere with each other constructively. A minimal degree of symbiosis requires that the actions of at least one agent A_i evolve so that they enable the fitness of at least one of the remaining agents A_j $(j \neq i)$ to increase above the fitness it achieved on its own. For each agent in the group, the total amount by which its fitness is increased by interaction must be at least as great as any decrease. (The net increase may be 0 for some agents).

Example of Symbiosis

To answer the question "why symbiosis" an example can be very helpful. To illustrate the type of symbiosis that can emerge between different perspectives, we may consider two robots A_1 and A_2 acting in a world containing two types of objects A and B. A_1 is a construction robot whose task is to search for objects of type A and construct as many components as possible using these objects. Each completed component must then be stacked in any position in the room. A_2 is a clearing-up robot whose task is to stack all free lying objects of any type in a corner in the room. This may be any corner and there may be one or more stacks. We will assume that individual objects are continually appearing at random positions (disorder is always being produced). Furthermore components produced by A_1 are continually being consumed. Objects placed far away take longer to fetch than nearer objects. Details of stacking or construction will not be considered here.

The fitness function for A_1 can be the average number of completed components stacked at position P during its time of operation which means that it should construct components as quickly as possible. For A_2 the fitness function could be the ratio of objects stacked in a corner to those lying around. It is of no interest to A_2 if these objects are assembled in components or not. Then there are several possible ways in which symbiosis can emerge:

A_1 can help A_2 as follows:

a) if there are any objects of type A lying around unstacked, use them first to construct a component; this reduces the amount of clearing up.

b) choose the stacking position to be a corner; this also does part of A_2's work since it only recognizes stacked objects, regardless of whether they are assembled in components or not.

A_2 can help A_1 as follows:

a) start collecting objects of type A first, only collect type B when there are none of type A.

b) stack them in the corner nearest to where A_1 is working.

Both these actions reduce the amount of time A_1 spends searching.

We can already make some interesting observations here. First, each agent represents a different perspective. For example, the "objects" recognized by A_1 are type A objects and the larger components made out of them; type B objects are of no importance (unless they act as obstacles). In contrast, A_2 does not recognise assembled components since they are not important in the fulfilment of its goal. So when A_1 will "perceive" a stack of components, A_2 will "perceive" a stack of individual objects. This means that we have a plurality or "redundancy" of representations for something which could be described as a single problem (i.e. construct components and keep the place tidy at the same time). This is where bottom-up evolution of requirements has advantages.

Secondly, because each agent at least partly does the work of the other, if one fails, then there is no catastrophic degradation in the satisfaction of requirements. We will show later that symbiosis of this kind can also help detection of destructive anomalies in the operational phase.

Initial Architecture

We can now consider an architecture for the above kind of problem. First, we define two important concepts:

a) A *perspective* is a representation of a particular goal, along with the relevant objects and actions for achieving it (world model). A goal-seeking agent represents a perspective (see example above).

b) Two agents A_1 and A_2 (and perspectives they represent) have *independent origins* if no aspect of A_1's functioning depends on explicitly encoded knowledge of A_2, i.e. they are initially "foreign" to each other.

We then require a kind of "mediation agent" M whose task it is to find possible forms of symbiosis between such independent agents, or at least establish "self-tolerance". Although M has approximately the same function as a "negative selection" algorithm, it should not be a global management component; it should itself be subject to monitoring and corrective modification by parts of the immune system which are already in the operational phase. However, for simplicity we will not consider this at present. M can be implemented as a co-evolutionary GP algorithm which takes the perspectives to be mediated as parameters.

Since agents have no predefined knowledge of each other and we wish to dispense with global management, interactions between agents have to be defined differently. Initially, any action of an agent A_1 which interferes with another agent A_2 will be detected as "disturbances" or anomalies by A_2 because they will not usually fit into its internal model (due to its independent origin).

A perspective-representing agent A_i can be implemented as a process which executes a given strategy (a gp), evaluates its fitness and detects anomalies. It is assumed that it has an internal model of the world, which is encoded according to its perspective. We will not consider evolution of this model or any kind of life-time learning at this stage. The gp that it executes is a candidate solution to a problem according to the perspective which the agent represents (e.g. construction of a component). A perspective can be implemented as a GP specification $< F, T, f >$ where F is the function set, T is the terminal set and f is the fitness function. Basically this defines the kind of objects in the world and the operations on them. For each perspective there is one population of gp's. Then for population size m and number of perspectives (populations) n, g_{ik} is the kth genetic program in the ith population:

$$\begin{array}{lllll} A_1: & g_{11} & g_{12} & \dots\dots & g_{1m} \\ A_2: & g_{21} & g_{22} & \dots\dots & g_{2m} \\ \dots & \dots\dots & \dots\dots & \dots\dots & \\ A_n: & g_{n1} & g_{n2} & \dots\dots & g_{nm} \end{array}$$

What this means is that an agent A_i can be "given" any of the strategies g_{ik} to use as part of its code.

Initially, during the maturation phase, this is done by M but later in the operational phase, it may be done by any other agent (see later).

An "individual" I_k undergoing evolution is defined as a multi-agent system, where each agent executes and evaluates one gp from each population. In other words, it is a candidate for symbiosis (or at least self-tolerance). As with general co-evolution, one agent acts as the "environment" for the other. The agents are activated as concurrent processes during the testing phase of each generation. The simplest variant is to have a single population with m individuals, where an individual is a column in the table, i.e.:

	I_1	I_2	......	I_m
A_1	g_{11}	g_{12}	......	g_{1m}
A_2	g_{21}	g_{22}	......	g_{2m}
...	......	.....	....	.....
A_n	g_{n1}	g_{n2}	......	g_{nm}

For the more general (and probably more effective) multi-population variant, a particular g_{ik} may be tested in different "environments" (i.e. in the presence of different gp's).

Perspective-Dependent Fitness

Fitness must be evaluated at the micro-level (that of a single gp) and at the macro-level (the whole network). When considering the micro-level, the "effective" fitness of a gp may be defined as the actual measure which is used to determine selection. For this measure, we wish to include the "supportive" actions of an agent for other tasks, in addition to its effectiveness at its own task. Then for two agents, there are four intermediate fitness values as shown in the table:

	A_1	A_2
A_1	F_{11}	F_{12}
A_2	F_{21}	F_{22}

For any two agents A_i and A_j, F_{ij} indicates the fitness measure of agent i from perspective j, i.e. the "support" of agent i for the task of agent j. These values depend on the detection of anomalies (disturbances) produced by the other agent. Details of such an anomaly-detecting agent are given elsewhere [7].

To show briefly how this works, we will consider what happens from perspective j. To determine F_{jj} the agent evaluates its own current gp j by calculating its effects according to its internal world model. The quality (beneficial or not) is then determined by the fitness function associated with perspective j. The same gp is then repeatedly executed in the current environment (i.e. in the presence of A_i which is running concurrently). An anomaly may then be detected as a difference between model-predicted and actual state, and is assumed to be due to A_i. The quality of this difference is evaluated according to the fitness function of A_j (is the effect helpful or disruptive?). Then the average effect of A_i according to A_j's fitness function is assigned to F_{ij}.

Rewarding Helpfulness

The effective fitness E_i used by the GP algorithm M to determine selection of a gp g_{ik} not only depends on F_{ii} but also on F_{ij}. This could be e.g.

$$E_i = F_{ii} + wF_{ij}$$

where w is the weight associated with effects on other agents. Then the fitness of an individual k may be calculated according to the degree to which the individual agents carry out their own tasks and the degree of cooperation based on the F_{ij} values where $j \neq i$.

Self-Nonself Discrimination

After the maturation phase, a self-definition should have evolved which can be defined as a symbiotic network of at least two agents. A distributed, multi-perspective form of self-nonself discrimination can now take place. What was previously "environment" from a particular perspective now becomes "self" from that perspective. This means that from the point of view of an agent A_1, "self" is A_2 and conversely for A_2, "self" is A_1. For A_1 the functional self (what is to be preserved) is the fitness F_{21} of A_2 which is calculated according to the above algorithm. This should be 0 or positive (self-tolerance or symbiosis); it should never be negative. The structural "self" relative to A_1 is the algorithm implementing A_2 (in particular its current gp). Self-reflection within a single agent may also be included but this is not relevant here.

In this architecture, the internal model of an agent does not itself evolve and this means that anomalies will continue to occur (because they continue to contradict a static internal model). However, in the operational phase, anomalies will either be harmless or constructive (because of symbiosis). Then for A_1, a sudden negative change in the "signature" of "self" (i.e. A_2) indicates that some external destructive agency is interfering with the functionality of A_2 as measured by the fitness evaluation. This requires an immune response. However, a harmless or constructive nonself is tolerated, since the functional self-definition is based on functionality (fitness function) instead of specific patterns. This avoids allergic reactions and makes additional symbiotic relationships possible.

Autonomous Recovery of Functionality
Since there is no life-time learning in this model, a gp cannot adapt directly to anomalies and recover its own functionality (self-reflection). However, autonomous recovery may still be possible with this architecture if the agents have some form of "modification access" to each other's code (effectively each agent carries a "benevolent virus"). At present, there is a lack of suitable software tools and programming languages which make this easier (although some work is being done here, e.g. Kaehr and Mahler [4]).

A first step in this direction is possible as follows: if A_1 detects a negative nonself anomaly, the functionality of A_2 must be recovered. To do this there is a whole population of gp's which also implement the functionality of A_2 although not optimally. If we ensure that there is enough variation in the population, one of them may have some features which can respond more effectively to the anomalous situation than the current gp. Then A_1 can take corrective action by selecting an alternative implementation from the population of gp's for A_2. The selection could take place as a form of parallel fitness evaluation. This could be implemented by stopping the current version of the process for A_2 and starting a different version with the new gp, and possibly alternative algorithms for the other functions of A_2 as well (i.e. its own anomaly-detection and fitness evaluation capabilities).

The greater the degree of symbiosis that has evolved, the more effective such a corrective action can be, because then the population should mostly consist of gp's which carry out their own function by constructively interfering with the function of other agents as well. This means that the fitness as seen from perspective 1 cannot differ too much from that of perspective 2 and a selection of the "best" gp for A_1 (highest F_{21}) will probably also be very good for A_2, even though no fitness evaluation for A_2 has yet taken place (i.e. F_{22}). It is not expected to be near optimal however, although subsequent improvements are possible if we introduce life-time learning and single-agent reflection.

4. DISCUSSION AND FUTURE WORK

In the architecture presented here, the "self" from the perspective of A_1 can still include parts of the algorithm of A_1 itself (in the sense of traditional self-reflection algorithms). However, because we insisted on the perspectives having independent origins, mutual reflection between agents is clearly more reliable than self-reflection within one agent. This becomes clear when we consider that in the case of single-agent reflection, a virus has only to disable a single piece of code (in particular, the mechanisms carrying out the reflection can be attacked) whereas in the case of mutual monitoring, an attack must be successful against two completely independent software components with no explicitly programmed "knowledge" of each other. There is therefore some progress in removing reflective residues, although clearly it is still far from the optimal solution at this stage.

Extensions and Improvements
Further progress in diminishing reflective residues will require work in the following areas:

a) Introduction of life-time learning: further adaptation of evolved gp's, in particular in response to anomalies.

b) Increasing the number of components that are subject to evolution; e.g. the internal model, the methods of detecting anomalies, strategies for life-time learning.

c) More complex network topologies for mutual reflection: we have only presented a minimal architecture of two agents. For three or more agents there are many more possible ways in which mutual reflection can take place.

d) Including the supporting infrastructure within the mutually reflective architecture, i.e. the evolutionary process (M) in the maturation phase along with any other supporting software (e.g. operating system) and hardware.

Problems Still to be Overcome
There are still some fundamental challenges. First, current evolutionary programming and automated design techniques do not allow for evolution of substantially complex software components. However this can be partly overcome by having a reserve of alternative designs for the same functionality (which can be autonomously replaced). Furthermore, evolu-

tionary techniques could be applied to selected components of hand-crafted software.

Secondly, software tools and programming languages are required so that independent programs can modify each other's code, the most important requirement being that they do not have destructive effects.

5. REFERENCES

[1] L. Bull and T.C. Fogarty, "Artificial Symbiogenesis", *Artificial Life*, Vol 2, No 3, 1996, pp. 269-292.

[2] D. Dasgupta and S. Forrest, "Novelty-Detection in Time Series Data using Ideas from Immunology", *Proceedings of the International Conference on Intelligent Systems, 1997*

[3] R. Kaehr, "Zur Logik der 'Second Order Cybernetics'", *Kybernetik und Systemtheorie - Wissenschaftsgebiete der Zukunft?*, edited by ICS (Institute for Cybernetics and Systems Theory), IKS-Berichte, Dresden, 1991.

[4] R. Kaehr and T. Mahler, "Introducing and Modelling Polycontextural Logics", *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, Vienna, 1996.

[5] P.Maes and D. Nardi (editors), *Meta-Level Architectures and Reflection*, North-Holland, 1988.

[6] Y. Ishida, "The Immune System as a Self-Identification Process: a Survey and a Proposal", *International Workshop on Immunity-Based Systems 1996*, held in conjunction with ICMAS 96.

[7] C. Kennedy, "A Conceptual Foundation for Autonomous Learning in Unforeseen Situations" *IEEE International Symposium on Intelligent Control ISIC 1998*

[8] M. Klein, "Conflict Resolution in Cooperative Design", *International Journal for Artificial Intelligence in Engineering*, Vol.4, no.4, 1990, pp. 168-180.

[9] J.R. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection*, MIT Press, Cambridge MA, 1992.

[10] Maturana, H. and Varela, F. (1980): *Autopoiesis and Cognition: The Realization of the Living.* D.Reidel Publishing Company, 1980.

[11] G. Spanoudakis and A. Finkelstein, "Reconciliation: Managing Interference in Software Development", *ECAI96 Workshop on Modelling Conflicts in AI*, 1996.

[12] Varela, F. (1979) *Principles of Biological Autonomy*, North-Holland, 1979.

[13] E. von Goldammer, C. Kennedy, J. Paul, H. Lerchner, R. Swik, "Autonomous Systems: Description and Construction", *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, Vienna, April 1996.

[14] E. von Goldammer, J. Paul, C. Kennedy, "Dead and Living Systems: Their Relation to Formal Logical Descriptions" in *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, Vienna, April 1996.