# Chapter 5

# The Artificial Lymphocyte Life Counter

*"As a first approximation, I define "belief" not as the object of believing but as the subject's investment in a proposition, the act of saying it and considering it as true."*

- Thomas Carlyle

This chapter explains how artificial lymphocytes (ALCs) cover the non-self space, how their receptors are initialised and how their *affinity distance thresholds* are trained. The discrimination between *self* and *non-self* space is defined. The life cycle of an ALC is discussed and the life counter threshold function is introduced. The life counter threshold function determines in which state an ALC is in its life cycle. The requirements and the training of an ALC to classify a pattern as *non-self*, are also explained.

## 5.1 Introduction

The natural immune system has mature or memory lymphocytes with receptors on each lymphocyte's surface that function as pattern detectors or classifiers. These receptors distinguish between *self* (normal cells) and *non-self* (antigens) patterns by only binding to *non-self* patterns, i.e. the mature or memory lymphocyte does not bind to any *self* (normal cell). The receptors bind to the *non-self* pattern with a certain affinity. A lymphocyte binding with a higher affinity than another indicates that the lymphocyte has a better detection of the *non-self* pattern. The task of the artificial lymphocyte (ALC) is to be able to classify any pattern as *self* or *non-self* by detecting only *non-self* patterns and leave known self patterns undetected. In the context of computational pattern recognition, *self*, $S$, represents all patterns in a finite space, $\mho$, that is legitimate in a non-biological environment and *non-self*, $\overline{S}$, represents all patterns that is not in *self*.

That is,

$$S \cup \overline{S} = \mho$$

For an ALC to detect a non-self pattern, the receptor of the ALC must match the non-self pattern with a certain affinity.

The purpose of the AIS is to create an optimal set of ALCs that can detect and classify *non-self* from *self* (without detecting *self*) and to remove ALCs that do not classify any patterns or have a low probability to detect any *non-self* patterns. The AIS creates ALCs by randomly initialising the receptors and sets the *affinity distance threshold* by using a known *self* set of patterns to guarantee that created ALCs do not detect any of the self patterns in the known *self* set. Usually the set of ALCs is static. The ALCs are randomly initialised and their receptors are trained with a training set that consists of *non-self* and *self* patterns using negative selection or positive selection, i.e. supervised learning. The negative selection method selects patterns that do not match *self*. The static set of trained ALCs is then tested to determine classification performance. A larger set of trained ALCs results in better classification and a too small set of trained ALCs result in worse classification. The unsupervised approach to train the ALCs with positive or negative selection also uses a static set of randomly initialised ALCs during training. A larger set of randomly initialised ALCs classifies the training set better than a smaller set of randomly initialised ALCs. The clusters formed by the ALCs are then tested with a test set. It is assumed in this dissertation that the receptor of an ALC is a binary vector and that all patterns that need to be classified are in binary format. It is also assumed that the AIS has a static incomplete *self* set for training. That is, incremental learning is not considered.

The rest of this chapter describes the architecture of the artificial lymphocyte (ALC). The sections to follow introduce the different states in the ALC's life cycle and discusses the different selection techniques to train an ALC. A threshold function, the *Life Counter* (*LC*), is also presented and the influence of the parameters on the function is explained. The threshold function determines the status of an artificial lymphocyte within the lymphocyte life cycle. The AIS uses the status of an ALC to determine if the ALC must be removed from the set of ALCs.

## 5.2 The Artificial Lymphocyte

As stated in section 5.1, the purpose of an ALC is to detect and classify non-self patterns from self patterns. Since the B-Cell has receptors on its surface to bind to antigen and the T-Cell has receptors on its surface to match viral proteins that are represented by MHC-molecules as peptides, both B-Cell and T-Cell lymphocytes are modeled into a general artificial lymphocyte with a bit-string receptor. The receptor has a length $k$, equal to the length of a self and non-self pattern. The receptor has an affinity-distance threshold (ADT) that must be met to detect non-self patterns. In other words, the affinity of an ALC to a pattern must be greater than or equal to ADT for a match to occur, in which case the pattern is considered as non-self. Section 5.2.3 explains the role of the ADT in detecting non-self patterns. The artificial lymphocyte's receptor cannot bind or match a non-self pattern with an affinity that is greater than the size of the pattern, therefore $ADT \leq k$. The receptor of an ALC is randomly initialised to represent any pattern in the search space. That is, before an ALC can detect non-self patterns, it needs to be trained not to detect self patterns. Training is done by measuring the receptor of an ALC against the static incomplete self set to determine the ALC's affinity-distance threshold, as discussed in section 5.2.2. This guarantees that the ALC will not overlap with any of the known self patterns, and that the ALC's initial status is mature. The ALC's status is determined by a threshold function called the *Life Counter* (*LC*), which is explained in section 5.3. The status of an ALC indicates whether the ALC must be replaced with a newly constructed or evolved ALC, or be given higher priority than other ALCs.

### 5.2.1 Life Cycle of the Artificial Lymphocyte

This section introduces and explains the different states an ALC can have during its life cycle. The status of an ALC, which can be memory, mature or annihilated, determines its priority in a group of ALCs. The status of a trained ALC can be one of three stages:

**Annihilated (Low priority):** An ALC with this status has either been mutated to detect a self pattern or it has not detected any patterns as non-self and is then declared as obsolete in the group of ALCs. Annihilated ALCs must be removed from the group of ALCs.

**Mature (Medium priority):** The mature-ALC detects non-self patterns on a regular basis. An ALC in the mature status will be demoted to annihilated status if the ALC starts to detect no non-self patterns. If the ALC starts to detect non-self patterns more frequently it will be promoted to

memory status again.

**Memory (High priority):** Memory-ALCs detect non-self patterns more frequently than mature-ALCs. Memory-ALCs are given higher priority than other ALCs by evaluating an incoming pattern first before other ALCs, resulting in a faster non-self detection. These ALCs are not mutated nor replaced by any newly constructed ALC. An ALC in memory status cannot be annihilated and first needs to be demoted to mature status. This happens when an ALC in memory status does not frequently detect non-self patterns as before, resulting in demotion to mature status.

An ALC with immature status has not yet been trained by the AIS and can therefore not be used to detect any non-self patterns. ALCs with annihilated status (low priority) have a higher probability to be removed from the set of ALCs than ALCs with mature status (medium priority). As long as an ALC has memory status (high priority), the ALC will never be removed from the set of ALCs, since the memory ALC has a higher probability in classifying a pattern than an ALC with mature status or even annihilated status. Therefore it is important to determine, at any time, which ALCs are in memory status. The LC threshold function is used to determine the status of an ALC, and to translate an ALC from one state to another (as is discussed in section 5.3).

### 5.2.2 Training the ALC to Cover Non-Self Space

An ALC can be trained with one of two selection methods: negative selection or positive selection. Both negative and positive selection methods determine the best ADT for the ALC. A non-self pattern can be incorrectly classified by an ALC as a self pattern. This misclassification is known as a *false negative*. A self pattern can also be incorrectly classified by an ALC as a non-self pattern. This misclassification is known as a *false positive*. For purposes of training, an incomplete static self set is used and the bit-string receptor is randomly initialised. The two selection methods are described next.

#### 5.2.2.1 Adapted Negative Selection

The adapted negative selection method (refer to Figure 5.1) trains an ALC to have a maximum affinity-distance threshold, $a_{neg}$, that does not overlap with the self set. All patterns with a hamming distance from the ALC that is less than $a_{neg}$, are detected as non-self. To guarantee a maximum $a_{neg}$ with no overlap with self, $a_{neg}$ is set to the hamming distance of the nearest self

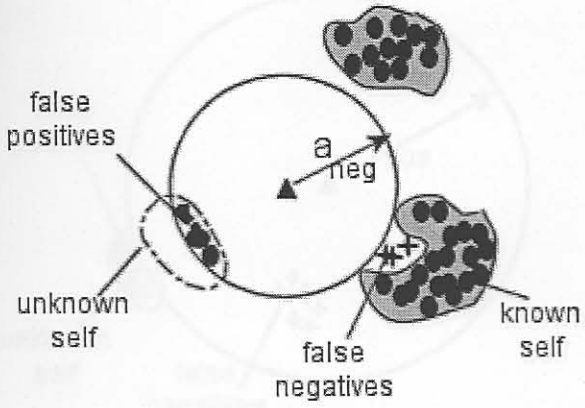


Figure 5.1: Venn-diagram of Adapted Negative Selection ALC

pattern to the ALC. Figure 5.1 shows that, with the adapted negative selection, the self pattern with the smallest hamming distance to the ALC is used to determine the maximum $a_{neg}$.

#### 5.2.2.2 Positive Selection

The positive selection method (refer to Figure 5.2) trains an ALC to have a minimum affinity-distance threshold, $a_{pos}$, that does overlap with the self set. All patterns with a hamming distance from the ALC that is greater than $a_{pos}$, are detected as non-self. To guarantee a minimum $a_{pos}$ with overlap with self, $a_{pos}$ is set to the hamming distance of the self pattern furthest from the ALC. Figure 5.2 shows that, with positive selection, the self pattern with the biggest hamming distance to the ALC is used to determine the minimum $a_{pos}$.

Figures 5.1 and 5.2 illustrate the drawback of *false positives* and *false negatives* when the respective selection methods train the ALCs. These drawbacks are due to an incomplete static self set. The *known self* is the incomplete static self set that is used to train the ALCs and the *unknown self* is the self patterns that are not known during training. The *unknown self* can also represent self patterns which are outliers to the set of *known self* patterns.

### 5.2.3 Matching a Non-Self Pattern

In the natural immune system, a B-Cell that regularly detects antigen goes into a memory status. This memory B-Cell is used in a secondary response to detect antigen faster than the primary response. It is therefore important to keep record of the number of non-self pattern matches (or

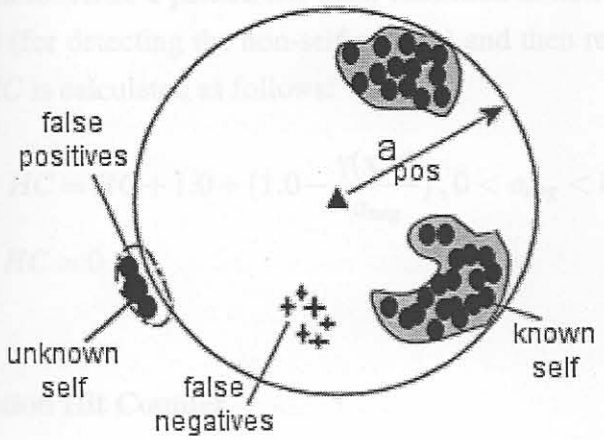


Figure 5.2: Venn-diagram of Positive Selection ALC

detections) made by an ALC to determine the ALC's matching ratio after classifying a number of non-self patterns. The *HitCounter* (*HC*) of an ALC is updated to keep record of the number of matches. The hamming distance between a pattern and the ALC receptor can be used to determine if the ALC detects a non-self pattern, defined as follows:

$$\gamma(\mathbf{x},\mathbf{r}) = \sum_{i=1}^{i<k} XOR(x_i, r_i)$$

where $\mathbf{x}$ is the bit-string of the pattern that needs to be classified, $\mathbf{r}$ is the bit-string of the receptor, *XOR* is the exclusive-or between the bits of the two bit-strings, $i$ is the bit-index and $k$ is the size of a pattern. Section 5.2.3.1 and Section 5.2.3.2 describe the requirements for an ALC to detect a non-self pattern and how the *HitCounter* of the ALC is updated for negative and positive selection respectively.

### 5.2.3.1 Adapted Negative Selection Hit Counter

As discussed in section 5.2.2.1, for an ALC trained using the adapted negative selection to detect a non-self pattern, the following relation must hold: $\gamma(\mathbf{x},\mathbf{r}) < a_{neg}$, i.e. the non-self pattern needs to be closer to the ALC than the closest self pattern to that ALC. A smaller value of $\gamma(\mathbf{x},\mathbf{r})$ indicates a more exact match to the pattern of the receptor. Then, $\frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$ is the difference ratio of a pattern with the receptor. The complement of $\frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$, which is $1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$, is the similarity ratio between a pattern and the receptor. The *HC* of an ALC that matches a non-self pattern more exact (i.e. $\gamma(\mathbf{x},\mathbf{r}) \to 0$, $1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}} \to 1.0$) than an ALC where $\gamma(\mathbf{x},\mathbf{r}) \to a_{neg}$, must be in-

cremented (rewarded) more. After a pattern has been classified as non-self, the $HC$ of the ALC is incremented with 1.0 (for detecting the non-self pattern) and then rewarded by the similarity ratio $1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$. The $HC$ is calculated as follows:

$$HC = HC + 1.0 + (1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}), 0 < a_{neg} < k$$

with the initial value of $HC = 0$.

#### 5.2.3.2 Positive Selection Hit Counter

As discussed in section 5.2.2.2, for an ALC trained using positive selection to detect a non-self pattern, the following relation must hold: $\gamma(\mathbf{x},\mathbf{r}) > a_{pos} \Rightarrow \gamma(\mathbf{x},\mathbf{r}) - a_{pos} > 0.0$, i.e. the non-self pattern needs to be further away from the ALC than the furthest self pattern to that ALC. A larger value of $\gamma(\mathbf{x},\mathbf{r})$ indicates a less exact match to the pattern of the receptor. Since $a_{pos}$ indicates the maximum ADT to a self pattern, the complement $k - a_{pos}$, indicates the maximum ADT to a non-self pattern. Then, $\frac{\gamma(\mathbf{x},\mathbf{r}) - a_{pos}}{k - a_{pos}}$ is the difference ratio of a pattern to the receptor. The $HC$ of an ALC that matches a non-self pattern less exact (i.e. $\gamma(\mathbf{x},\mathbf{r}) \rightarrow k$) than an ALC where $\gamma(\mathbf{x},\mathbf{r}) \rightarrow a_{pos}$, is incremented (rewarded) more. After a pattern has been classified as non-self, the $HC$ of an ALC is incremented with 1.0 (for detecting the non-self pattern) and then rewarded by the difference ratio $\frac{\gamma(\mathbf{x},\mathbf{r}) - a_{pos}}{k - a_{pos}}$. The $HC$ is calculated as follows for positive selection:

$$HC = HC + 1.0 + (\frac{\gamma(\mathbf{x},\mathbf{r}) - a_{pos}}{k - a_{pos}}), 0 < a_{pos} < k$$

with the initial value of $HC = 0$.

#### 5.2.3.3 The Hit Ratio

The *HitRatio* ($HR$) calculates the ratio at which an ALC matched non-self patterns. The hit ratio of an ALC is calculated after a specified number of patterns has been classified, the *IterationSize* ($IS$). A *HitRatio* ($HR$) is then calculated with parameters $HC$ and $IS$ as follows:

$$HR(HC, IS) = \frac{HC}{IS}$$

The HR is calculated to determine an ALC's detection ratio of non-self patterns in an iteration, i.e. the number of non-self patterns that were detected or matched by an ALC during an iteration.

## 5.3 The Life Counter

As stated earlier, the life counter (*LC*) determines in which state an ALC is at any given time. There are three states in which an ALC can be, prioritised into low, medium and high (as described in section 5.2.1). The *LC* is used to map the states to a continuous range $(0,1)$ (i.e. $m(LC) : \{low, medium, high\} \rightarrow (0,1)$). A life counter value closer to 1.0 indicates that the ALC has a higher priority than an ALC with a life counter value close to 0.0. The LC value of an ALC is calculated at specified time steps. The time step can be set to a constant iteration size (*IS*) of incoming patterns. The value of the LC depends on the ALC's *HR*, the *minimum matching ratio* (coverage of non-self space) of an ALC, and the *IS*.

For example, consider an ALC $A$ with $a = 10$ and ALC $B$ with $a = 5$ (assume that both $A$ and $B$ are initialised with the adapted negative selection). Let $k = 15$. $A$ covers more non-self space than $B$ since $a_A > a_B$. Therefore, it can be expected that $A$ must have a higher *HR* than $B$. Suppose $B$ has a higher *HR* than $A$, then $B$ has a higher probability to gain memory status than $A$. For A to have the same status as B, $HR_A = 2 * HR_B$ since $a_A = 2 * a_B$. Therefore, an ALC converges to memory status if the number of classified non-self patterns is greater than the number of patterns in the non-self space covered by the ALC. An ALC converges to annihilated status if the number of classified non-self patterns is less than the number of patterns in the non-self space covered by the ALC.

The minimum matching ratio of an ALC to detect a non-self pattern also influences the status of the ALC. The minimum matching ratio, $\beta$, is calculated as follows:

- For an ALC trained with the adapted negative selection,

$$\beta_{neg} = 1.0 - \frac{a_{neg}}{k}$$

- For an ALC trained with positive selection,

$$\beta_{pos} = \frac{a_{pos}}{k}$$

After each *IS* patterns, the ALC's life counter is re-calculated to determine the ALC's state using the life counter threshold function (as explained below).

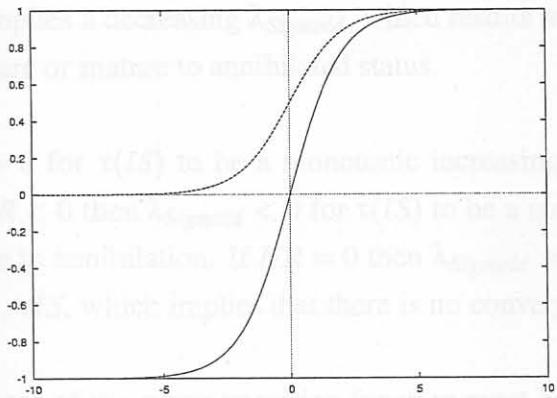


Figure 5.3: Hyperbolic Tangent Function and Sigmoid Function

### 5.3.1 Life Counter Threshold Function

The LC threshold function, $\tau$, determines the status of an ALC based on the $HR$, $\beta$ and $IS$ of the ALC. The $\tau$-value of the ALC is calculated after a number of patterns, $IS$, has been classified. The requirement for $\tau$ is that $\tau \in (0,1)$, thus a function with range of $(0,1)$ is needed, i.e.

$$\tau(IS) \rightarrow (0,1)$$

$\tau(IS)$ must be a continuous function and monotonic increasing from mature to memory status, and monotonic decreasing from memory to mature or mature to annihilation. The sigmoid function (dashed line in Figure 5.3),

$$g(\lambda_{Sigmoid}, x_{Sigmoid}) = \frac{1}{1+e^{(-\lambda_{Sigmoid} \times x_{Sigmoid})}},$$

where $\lambda_{Sigmoid}$ controls the steepness of $g(\lambda_{Sigmoid}, x_{Sigmoid})$, satisfies these requirements. For $g(\lambda_{Sigmoid}, x_{Sigmoid})$ to be a monotonic increasing function, $\lambda_{Sigmoid} > 0$ and for $g(\lambda_{Sigmoid}, x_{Sigmoid})$ to be a monotonic decreasing function, $\lambda_{Sigmoid} < 0$. Thus,

$$\tau(IS) = g(\lambda_{Sigmoid}, IS)$$

The steepness of $g(\lambda_{Sigmoid}, IS)$ represents the rate at which an ALC must converge to memory or annihilated status. As explained in section 5.2.3.3, the $HR$ of an ALC calculates the ratio at which an ALC matched non-self patterns. Thus, the $HR$ of an ALC determines the convergence ratio at which an ALC must converge to memory or annihilated status. An increasing $HR$ implies an increasing $\lambda_{Sigmoid}$, which results in faster convergence of an ALC from mature to memory

status. A decreasing $HR$ implies a decreasing $\lambda_{Sigmoid}$, which results in faster convergence of an ALC from memory to mature or mature to annihilated status.

If $HR > 0$ then $\lambda_{Sigmoid} > 0$ for $\tau(IS)$ to be a monotonic increasing function, implying convergence to memory. If $HR < 0$ then $\lambda_{Sigmoid} < 0$ for $\tau(IS)$ to be a monotonic decreasing function, implying convergence to annihilation. If $HR = 0$ then $\lambda_{Sigmoid} = 0$ for $\tau(IS)$ to be a linear function, with $\tau(IS) = 0.5, \forall IS$, which implies that there is no convergence to memory or annihilation.

To achieve this, the steepness of the status transition function must be a function of $HR$ with a range of $(-1,1)$. The rate at which an ALC's status moves from one state to the next is influenced by the minimum matching ratio $\beta$. For increasing $\beta$ the rate of a state transition should be faster than for a smaller $\beta$. A faster transition rate is obtained with a steeper gradient of the state transition function, which is achieved with $\lambda_{Sigmoid} > 0$. On the other hand, a slower transition (for smaller $\beta$) is obtained by using a $\lambda_{Sigmoid} < 0$. To achieve these effects, $\lambda_{Sigmoid}$ can be a function of the hyperbolic tangent function. The hyperbolic tangent function (solid line in Figure 5.3), is defined as

$$f(\lambda_{Hyper}, x_{Hyper}) = \frac{2}{1+e^{(-\lambda_{Hyper} \times x_{Hyper})}} - 1$$

where $\lambda_{Hyper}$ controls the steepness of $f(\lambda_{Hyper}, x_{Hyper})$. Thus, $\lambda_{Sigmoid}$ can be defined as

$$\begin{aligned} \lambda_{Sigmoid} &= f(\beta, HR) \\ &= \frac{2}{1+e^{(-\beta \times HR)}} - 1 \end{aligned}$$

In the above, $\beta$ controls the steepness of the hyperbolic tangent function. A large value of $\beta$ causes a higher rate of change of $f(\beta, HR)$, which ensures an increasing slope for the state transition function $g$. A smaller value of $\beta$ causes a lower rate of change of $f(\beta, HR)$, which ensures a decreasing slope for the state transition function. Values of $HR > 0$ will cause a positive output for $f(\beta, HR)$, which ensures a monotonic increasing state transition function. On the other hand $HR < 0$ will cause a negative output for $f(\beta, HR)$, which ensures a monotonic decreasing state transition function.

In summary:

$$\begin{aligned} \tau(IS) &\doteq g(\lambda_{Sigmoid}, IS) \\ &= g(f(\beta, HR), IS) \end{aligned}$$

$$= \frac{1}{1+e^{(-f(\beta,HR)\times IS)}}$$

where $f(\beta,HR) = \frac{2}{1+e^{(-\beta\times HR)}} - 1$

The $HR$ defined in section 5.2.3.3 will always have a positive value, since $IS > 0$ and $HC \geq 0$. With $HC \geq 0$ and $\beta \geq 0$, $f(\beta,HR) \geq 0$. This results in the fact that $g$ will always be a monotonic increasing function. This means that the function $g$ will always converge to 1.0 and all ALCs will either stay in the mature status or transolve to the memory status. To prevent all the ALCs to converge to memory, a constant detection ratio can be subtracted from $HR$. The constant detection ratio is called the expected matching ratio ($EMR$) and is explained in the following section.

## 5.4 The Expected Matching Ratio

The expected matching ratio ($EMR$) is the detection ratio of non-self patterns expected from an ALC. The $EMR$ needs to be updated so that after each iteration, the $EMR$ represents a more correct rate of non-self patterns that can be expected to be detected by an ALC. If $\eta_h$ is the number of non-self patterns detected in the current iteration, then $\frac{\eta_h}{IS}$ calculates the matching rate of non-self patterns per iteration. A proposed function to update the $EMR$ for the next iteration is to take the average over the current matching ratio and the current iteration's $EMR$, i.e.

$$EMR_{h+1} = \frac{EMR_h + \frac{\eta_h}{IS}}{2.0}$$

where $EMR_{h+1}$ is the calculated expected matching ratio for the next iteration. The $HR$ function is redefined to be also dependent on the $EMR$:

$$HR(HC,IS,EMR) = \frac{HC}{IS} - EMR$$

From the above equation it can be concluded that if an ALC detects less patterns as expected ($EMR > \frac{HC}{IS}$) then $HR < 0$, resulting in a monotonic decreasing $g$. If $EMR < \frac{HC}{IS}$ then $HR > 0$, which results in a monotonic increasing $g$. With the redefined $HR$, not all the ALCs will converge to memory, and those that match less non-self patterns as expected, move toward annihilated status.

## 5.5 Conclusion

This chapter explained how ALCs cover non-self space and how their receptors are initialised. The training of an ALC, using the adapted negative selection or positive selection, was explained. The requirements for classifying a non-self pattern were also presented in terms of the hit ratio function. The three types of status were prioritised into low, medium and high. Then the life counter threshold function was presented and explained. In the next chapter, the use of evolutionary computation techniques to evolve ALCs are discussed.