The FARB

In this chapter, we formally define the main tool developed in this work: the *fuzzy all-permutations rule-base* (FARB). We show that the special structure of the FARB implies that its IO mapping can be described using a relatively simple closed-form formula.

To motivate the definition of the FARB, we first consider a simple example adapted from [106; 107].

**Example 2.1.** Consider the following four-rule FRB:

$R_1$: If $x_1$ is *smaller than* 5 and $x_2$ *equals* 1, Then $f = -4$,

$R_2$: If $x_1$ is *larger than* 5 and $x_2$ *equals* 1, Then $f = 0$,

$R_3$: If $x_1$ is *smaller than* 5 and $x_2$ *equals* 7, Then $f = 2$,

$R_4$: If $x_1$ is *larger than* 5 and $x_2$ *equals* 7, Then $f = 6$.

Assume that the pair of terms *equals 1, equals 7* in the rules are modeled using the Gaussian membership function (MF):

$$
\mu_{=k}(y) := \exp \left( -\frac{(y-k)^2}{2\sigma^2} \right),
$$

with $k = 1$ and $k = 7$, and that the terms *larger than k, smaller than k* are modeled using the Logistic functions:

$$
\mu_{>k}(y) := \frac{1}{1 + \exp (-\alpha(y-k))} \quad \text{and} \quad \mu_{<k}(y) := \frac{1}{1 + \exp(\alpha(y-k))},
$$

where $\alpha$ and $\sigma$ are parameters.
with $\alpha > 0$ (see Fig. 2.1).

Applying the product-inference rule, singleton fuzzifier, and the center of gravity (COG) defuzzifier [164] to this rule-base yields: $f(x) = u(x)/d(x)$, where

$$u(x) = -4\mu_{<5}(x_1)\mu_{=1}(x_2) + 2\mu_{<5}(x_1)\mu_{=7}(x_2) + 6\mu_{>5}(x_1)\mu_{=7}(x_2),$$

$$d(x) = \mu_{<5}(x_1)\mu_{=1}(x_2) + \mu_{>5}(x_1)\mu_{=1}(x_2) + \mu_{<5}(x_1)\mu_{=7}(x_2)$$

$$+ \mu_{>5}(x_1)\mu_{=7}(x_2).$$

(2.2)

Rewriting $u(x)$ as

$$u(x) = (1 - 2 - 3)\mu_{<5}(x_1)\mu_{=1}(x_2) + (1 + 2 - 3)\mu_{>5}(x_1)\mu_{=1}(x_2)$$

$$+ (1 - 2 + 3)\mu_{<5}(x_1)\mu_{=7}(x_2) + (1 + 2 + 3)\mu_{>5}(x_1)\mu_{=7}(x_2),$$

and using (2.2) yields

$$f(x) = \frac{u(x)}{d(x)}$$

$$= 1 + 2\frac{(\mu_{>5}(x_1) - \mu_{<5}(x_1))(\mu_{=1}(x_2) + \mu_{=7}(x_2))}{(\mu_{>5}(x_1) + \mu_{<5}(x_1))(\mu_{=1}(x_2) + \mu_{=7}(x_2))}$$

$$+ 3\frac{(\mu_{>5}(x_1) + \mu_{<5}(x_1))(-\mu_{=1}(x_2) + \mu_{=7}(x_2))}{(\mu_{>5}(x_1) + \mu_{<5}(x_1))(\mu_{=1}(x_2) + \mu_{=7}(x_2))}.$$
A direct calculation shows that for our MFs:

\[
\frac{\mu_{>k}(y) - \mu_{<k}(y)}{\mu_{>k}(y) + \mu_{<k}(y)} = \tanh(\frac{\alpha(y - k)}{2}),
\]

\[
\frac{\mu_{=a}(y) - \mu_{=b}(y)}{\mu_{=a}(y) + \mu_{=b}(y)} = \tanh(\frac{(2y - a - b)(a - b)}{4\sigma^2}),
\]

so

\[
f(x) = 1 + 2\tanh((x_1 - 5)\alpha/2) + 3\tanh((x_2 - 4)/\sigma^2).
\]

Thus, the IO mapping \((x_1, x_2) \rightarrow f(x_1, x_2)\) of the FRB is \textit{mathematically equivalent} to the IO mapping of a feedforward ANN with two hidden neurons (see Fig. 2.2). Conversely, the ANN depicted in Fig. 2.2 is mathematically equivalent to the aforementioned FRB. Note that the network parameters are directly related to the parameters of the FRB, and vice versa. \(\square\)

\section*{2.1 Definition}

We say that a function \(g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}\) is \textit{sigmoid} if \(g\) is continuous, and the limits \(\lim_{y \rightarrow -\infty} g(y)\) and \(\lim_{y \rightarrow +\infty} g(y)\) exist. Example 2.1 motivates the search for an FRB whose IO mapping is equivalent to a linear combination of sigmoid functions, as this is the mapping of an ANN...
with a single hidden layer. This is the *FARB*. For the sake of simplicity, we consider a FARBR with output \( f(t) \); the generalization to the case of multiple outputs is straightforward.

**Definition 2.2.** An FRB with time-varying inputs \( x_1(t), \ldots, x_m(t) \) and output \( f(t) \) is called a FARBR if the following three conditions hold.

1. Every input variable \( x_i(t) \) is characterized by two verbal terms, say, \( \text{term}_i^- \) and \( \text{term}_i^+ \). These terms are modeled using two membership functions (MFs): \( \mu_i^- (\cdot) \) and \( \mu_i^+ (\cdot) \). Define
   \[
   \beta_i(y) := \frac{\mu_i^+(y) - \mu_i^-(y)}{\mu_i^+(y) + \mu_i^-(y)}.
   \]
   The MFs satisfy the following constraint: there exist sigmoid functions \( g_i(\cdot) : \mathbb{R} \to \mathbb{R} \), and \( q_i, r_i, u_i, v_i \in \mathbb{R} \) such that
   \[
   \beta_i(y) = q_i g_i(u_i y - v_i) + r_i, \quad \text{for all } y \in \mathbb{R}. \tag{2.3}
   \]

2. The rule-base contains \( 2^m \) fuzzy rules spanning, in their If-part, all the possible verbal assignments of the \( m \) input variables.

3. There exist \( a_i(t) : \mathbb{R} \to \mathbb{R}, i = 0, 1, \ldots, m, \) such that the Then-part of each rule is a combination of these functions. Specifically, the rules are:

\[
R_1 : \text{If } (x_1(t) \text{ is term}_1^-) \& (x_2(t) \text{ is term}_2^-) \& \ldots \& (x_m(t) \text{ is term}_m^-) \\
\text{Then } f(t) = a_0(t) - a_1(t) - a_2(t) - \cdots - a_m(t),
\]

\[
R_2 : \text{If } (x_1(t) \text{ is term}_1^+) \& (x_2(t) \text{ is term}_2^+) \& \ldots \& (x_m(t) \text{ is term}_m^+) \\
\text{Then } f(t) = a_0(t) + a_1(t) - a_2(t) - \cdots - a_m(t),
\]

\[
\vdots
\]

\[
R_{2^m} : \text{If } (x_1(t) \text{ is term}_1^+) \& (x_2(t) \text{ is term}_2^+) \& \ldots \& (x_m(t) \text{ is term}_m^+) \\
\text{Then } f(t) = a_0(t) + a_1(t) + a_2(t) + \cdots + a_m(t), \tag{2.4}
\]

where \( \& \) denotes “and”. Note that the signs in the Then-part are determined in the following manner: if the term characterizing \( x_i(t) \) in
the If-part is term$_{+,i}$, then in the Then-part, $a_i(t)$ is preceded by a plus sign; otherwise, $a_i(t)$ is preceded by a minus sign.

Summarizing, the FARB is a standard FRB satisfying several additional constraints: each input variable is characterized by two verbal terms; the terms are modeled using MFs that satisfy (2.3); the rule-base contains exactly $2^m$ rules; and the values in the Then-part of the rules are not independent, but rather they are a linear combination of the $m + 1$ functions $a_0(t), \ldots, a_m(t)$.

As we will see below, the IO mapping of the FARB is a weighted sum of the $g_i$s. We will be particularly interested in the case where each $g_i$ is a function that is commonly used as an activation function in ANNs (e.g., the hyperbolic tangent function, the Logistic function).

Remark 2.3. It is easy to verify that the FRB defined in Example 2.1 is a FARB with $m = 2$, $a_0(t) \equiv 1$, $a_1(t) \equiv 2$, and $a_2(t) \equiv 3$.

Remark 2.4. It may seem that the constraint (2.3) is very restrictive. In fact, several MFs that are commonly used in FRBs satisfy (2.3). Relevant examples include the following.

1. If the terms \( \{\text{term}_+, \text{term}_-\} \) are \( \{\text{equals } k_1, \text{equals } k_2\} \), respectively, where the term \( \text{equals } k \) is modeled using the Gaussian MF (2.1), then it is easy to verify that

\[
\beta(y) = \tanh(ay - b),
\]

with

\[
a := (k_1 - k_2)/(2\sigma^2),
\]

\[
b := (k_1^2 - k_2^2)/(4\sigma^2).
\]

Thus, Eq. (2.3) holds with

\[
g(z) = \tanh(z), \quad u = a, \quad v = b, \quad q = 1, \quad \text{and} \quad r = 0.
\]
Note that (2.5) can also be written as
\[
\beta(y) = 2\sigma(2ay - 2b) - 1,
\]
where \(\sigma(\cdot)\) is the \textit{logistic function}:
\[
\sigma(z) := (1 + \exp(-z))^{-1}.
\]
Thus, Eq. (2.3) holds with
\[
g(z) = \sigma(z), \quad u = 2a, \quad v = 2b, \quad q = 2, \quad \text{and} \quad r = -1.
\]

2. If the two MFs satisfy
\[
\mu_-(y) = 1 - \mu_+(y), \quad (2.6)
\]
(a common choice for two contradictory fuzzy terms), then \(\beta(y) = 2\mu_+(y) - 1\), so (2.3) holds. Interesting special cases include:

a) If the terms \{term\_+, term\_-\} are \{equals \(k\), not equals \(k\}\}, respectively, modeled using the MFs:
\[
\mu_{=k}(y) := \exp\left(-\frac{(y-k)^2}{2\sigma^2}\right), \quad \mu_{\neq k}(y) := 1 - \mu_{=k}(y),
\]
then
\[
\beta(y) = 2\exp\left(-\frac{(y-k)^2}{2\sigma^2}\right) - 1.
\]
Thus, (2.3) holds with
\[
g(z) = \exp(-z^2), \quad u = \sqrt{\frac{1}{2\sigma^2}}, \quad v = \sqrt{\frac{k^2}{2\sigma^2}}, \quad q = 2, \quad \text{and} \quad r = -1.
\]

b) If the terms \{term\_+, term\_-\} are \{larger than \(k\), smaller than \(k\}\}, respectively, modeled using the Logistic functions:
\[
\mu_{>k}(y) := \sigma(\alpha(y-k)), \quad \mu_{<k}(y) := \sigma(-\alpha(y-k)), \quad (2.7)
\]
with $\alpha > 0$, then

$$\beta(y) = 2\sigma(\alpha(y - k)) - 1.$$  \hfill (2.8)

Thus, (2.3) holds with

$$g(z) = \sigma(z), \ u = \alpha, \ v = \alpha k, \ q = 2, \ and \ r = -1.$$  \hfill (2.9)

Note that (2.8) can also be written as

$$\beta(y) = \tanh(\alpha(y - k)/2),$$

which implies that (2.3) holds with

$$g(z) = \tanh(z), \ u = \alpha/2, \ v = \alpha k/2, \ q = 1, \ and \ r = 0.$$  

c) If the terms \{term\_+, term\_-\} are \{positive, negative\}, respectively, modeled using:

$$\mu_{pos}(y) := \begin{cases} 0, & \text{if } -\infty < y < -\Delta, \\ (1 + y/\Delta)/2, & \text{if } -\Delta \leq y \leq \Delta, \\ 1, & \text{if } \Delta < y < \infty, \end{cases}$$

$$\mu_{neg}(y) := 1 - \mu_{pos}(y),$$  \hfill (2.10)

with $\Delta > 0$, then

$$\beta(y) = 2\sigma_{L}(y/(2\Delta) + 1/2) - 1,$$

where $\sigma_{L}(\cdot)$ is the standard piecewise linear Logistic function:

$$\sigma_{L}(y) := \begin{cases} 0, & \text{if } -\infty < y < 0, \\ y, & \text{if } 0 \leq y \leq 1, \\ 1, & \text{if } 1 < y < \infty. \end{cases}$$  \hfill (2.11)

\footnote{Note that the MFs in (2.7) satisfy (2.6).}
This implies that \((2.3)\) holds with

\[
g(z) = \sigma_L(z), \ u = 1/(2\Delta), \ v = -1/2, \ q = 2, \ \text{and} \ r = -1.
\]

d) If the terms \(\{\text{term}_+, \text{term}_-\}\) are \{\text{larger than} \(k\), \text{smaller than} \(k\)\}, modeled using:

\[
\mu_{>k}(y) := \mu_{\text{pos}}(y-k), \ \text{and} \ \mu_{<k}(y) := \mu_{\text{neg}}(y-k), \quad (2.12)
\]

with \(\mu_{\text{pos}}, \mu_{\text{neg}}\) defined in \((2.10)\), then

\[
\beta(y) = 2\sigma_L((y-k)/(2\Delta) + 1/2) - 1, \quad (2.13)
\]

so \((2.3)\) holds with

\[
g(z) = \sigma_L(z), \ u = 1/(2\Delta), \ v = \frac{k - \Delta}{2\Delta}, \ q = 2, \ \text{and} \ r = -1.
\]

Summarizing, all the above verbal terms can be modeled using MFs that satisfy \((2.3)\).

2.2 Input-Output Mapping

The next result provides a closed-form formula for the IO mapping of the FARB.

**Theorem 2.5.** Applying the product-inference rule, singleton fuzzifier, and the COG defuzzifier to a FARB yields the output:

\[
f = a_0(t) + \sum_{i=1}^{m} r_i a_i(t) + \sum_{i=1}^{m} q_i a_i(t) g_i(u_i x_i(t) - v_i). \quad (2.14)
\]

**Proof.** For the sake of notational convenience, we omit from hereon the dependence of the variables on \(t\). Definition 2.2 implies that inferring the FARB yields \(f(x) = u(x)/d(x)\), with
u(x) := (a_0 + a_1 + a_2 + \ldots + a_m)\mu^1_+(x_1)\mu^2_+(x_2)\ldots\mu^m_+(x_m) + (a_0 - a_1 + a_2 + \ldots + a_m)\mu^1_-(x_1)\mu^2_+(x_2)\ldots\mu^m_+(x_m) + \ldots + (a_0 - a_1 - a_2 - \ldots - a_m)\mu^1_-(x_1)\mu^2_-(x_2)\ldots\mu^m_-(x_m),

and

d(x) := \mu^1_+(x_1)\mu^2_+(x_2)\ldots\mu^m_+(x_m) + \mu^1_+(x_1)\mu^2_+(x_2)\ldots\mu^m_+(x_m) + \ldots + \mu^1_-(x_1)\mu^2_-(x_2)\ldots\mu^m_-(x_m),

where both u and d include $2^m$ terms.

Let

$$p(x) := a_0 + \sum_{i=1}^{m} a_i\beta_i(x_i). \quad (2.15)$$

It is easy to verify, by expanding the sum, that $p = u/d$. Thus, $f = p$.

Eqs (2.3) and (2.15) yield $p(x) = a_0 + \sum_{i=1}^{m} a_i(g_i(u_ix_i - v_i) + r_i)$, and this completes the proof.

Eq. (2.14) implies that the FARB output $f$ can be obtained by first feeding the (scaled and biased) inputs $u_ix_i(t) - v_i$ to a layer of units computing the activation functions $g_i(\cdot)$, and then computing a weighted (and biased) sum of the units outputs (see Fig. 2.3). Applications of
this resemblance between FARBs and ANNs for knowledge-based neurocomputing are presented in the following chapter.
The FARB–ANN Equivalence

In this chapter, we consider several special cases of the FARB. In each of these cases, the IO mapping of the FARB is mathematically equivalent to that of a specific type of ANN. This provides a symbolic representation of the ANN functioning. As a corollary, we obtain results that can be used for KE from, and KBD of, ANNs. Simple examples are used to demonstrate the main ideas. The remainder of this work is devoted to extending this approach to larger-scale networks.

3.1 The FARB and Feedforward ANNs

Consider a feedforward ANN with: inputs $z_1, \ldots, z_k$, $n$ hidden neurons with activation function $h(\cdot)$, weights $w_{ij}$ and a single output $o$ (see Fig. 3.1). For notational convenience, denote $w_{i0} = b_i$, and $y_i := \sum_{j=1}^{k} w_{ij} z_j$. Then

$$o = c_0 + \sum_{i=1}^{n} c_i h(y_i + w_{i0}).$$  \hspace{1cm} (3.1)

Consider (2.14) in the particular case where $x_i$ and $a_i$ are time-invariant, so,

\footnote{Note that in this special case, the FARB reduces to the APFRB defined in [90; 91].}
Comparing (3.1) and (3.2) yields the following results. We use the notation $[j:k]$ for the set \{j, j + 1, ..., k\}.

**Corollary 3.1. (KE from Feedforward ANNs)**

Consider the feedforward ANN (3.1). Let $f$ denote the output of a FARB with: MFs that satisfy (2.3) with $g_i = h$, $m = n$ inputs $x_i = y_i/u_i$, and parameters $v_i = -w_{i0}$, $a_i = c_i/q_i$ for $i \in [1:n]$, $a_0 = c_0 - \sum_{i=1}^{n} c_i r_i / q_i$.

Then $f = o$.

Corollary 3.1 implies that given a feedforward ANN in the form (3.1), we can immediately design a FARB whose IO mapping is mathematically equivalent to that of the ANN. This provides a symbolic representation of the ANN’s IO mapping. The next example demonstrates this.
3.1.1 Example 1: Knowledge Extraction from a Feedforward ANN

Consider the two-input-one-output ANN depicted in Fig. 3.2. The ANN output is given by

\[ o = \sigma(4z_1 + 4z_2 - 2) - \sigma(4z_1 + 4z_2 - 6). \]  \hspace{1cm} (3.3)

We assume that the inputs are binary, that is, \( z_i \in \{0, 1\} \), and declare the ANN decision to be one if \( o > 1/2 \), and zero otherwise. Unfortunately, both Fig. 3.2 and Eq. (3.3) provide little insight as to what the ANN is actually computing. In this simple case, however, we can easily calculate \( f \) for the four possible input combinations, and find that \( o > 1/2 \) if and only if (iff) \((z_1 \text{ xor } z_2) = 1\), so the ANN is computing the xor function.

We now transform this ANN into an equivalent FARB. Denote \( y_i := 4z_1 + 4z_2, i = 1, 2 \), that is, the inputs to the hidden neurons, and rewrite (3.3) as

\[ o = \sigma(y_1 - 2) - \sigma(y_2 - 6). \]

Note that this is in the form (3.1) with

\[ n = 2, \ c_0 = 0, \ c_1 = 1, \ c_2 = -1, \ w_{10} = -2, \ w_{20} = -6, \text{ and } h = \sigma. \]
We apply Corollary 3.1 to design a FARB with \( m = n = 2 \) inputs \( x_i \), and IO mapping \( (x_1, x_2) \rightarrow f(x_1, x_2) \) that is equivalent to the mapping \( (y_1, y_2) \rightarrow o(y_1, y_2) \). Suppose that for input \( x_i, i = 1, 2 \), we choose to use the verbal terms \textit{larger than} \( k_i \) and \textit{smaller than} \( k_i \), modeled using (2.7) with, say, \( \alpha = 4 \). Then (2.8) yields

\[
\beta(y_i) = 2\sigma(4(y_i - k_i)) - 1,
\]

so the parameters in (2.3) are

\[
q_i = 2, \ g_i = \sigma, \ u_i = 4, \ v_i = 4k_i, \ r_i = -1.
\] (3.4)

Applying Corollary 3.1 implies that the equivalent FARB has inputs \( x_1 = y/4, x_2 = y/4 \), and parameters: \( a_0 = 0, a_1 = 1/2, a_2 = -1/2, \)
\( v_1 = 2, \) and \( v_2 = 6, \) so (3.4) yields \( k_1 = 1/2 \) and \( k_2 = 3/2 \). Summarizing, the equivalent FARB is:

\[ R_1: \text{If } y/4 \text{ is smaller than } 1/2 \text{ & } y/4 \text{ is smaller than } 3/2, \text{ Then } f = 0, \]
\[ R_2: \text{If } y/4 \text{ is smaller than } 1/2 \text{ & } y/4 \text{ is larger than } 3/2, \text{ Then } f = -1, \]
\[ R_3: \text{If } y/4 \text{ is larger than } 1/2 \text{ & } y/4 \text{ is smaller than } 3/2, \text{ Then } f = 1, \]
\[ R_4: \text{If } y/4 \text{ is larger than } 1/2 \text{ & } y/4 \text{ is larger than } 3/2, \text{ Then } f = 0, \]

where ‘&’ denotes ‘and’.

This FARB provides a \textit{symbolic} description of the ANN’s IO mapping. It can be further simplified as follows. Rule \( R_2 \) is self-contradicting and can be deleted. The remaining three rules can be summarized as:

If \( z_1 + z_2 \) is larger than 1/2 and smaller than 3/2, Then \( f = 1; \)
Else \( f = 0. \)

Recalling that \( z_i \in \{0, 1\} \), we see that this single rule is indeed an intuitive description of the function \( f(z_1, z_2) = z_1 \text{ xor } z_2 \). Thus, the transformation from an ANN to an equivalent FARB yields a comprehensible representation of the network operation. \( \square \)
The next result is the converse of Corollary 3.1, namely, it states that given a FAR, we can represent its IO mapping in the form of an ANN.

**Corollary 3.2. (KBD of Feedforward ANNs)**

Consider a FAR with \( m \) inputs \( x_1, \ldots, x_m \) and output \( f \). Suppose that (2.3) holds for all \( i \in [1, m] \) such that \( g_1 = \cdots = g_m \). Define \( n = m \), \( y_i = u_i x_i \), \( w_{i0} = -v_i \), \( c_i = a_i q_i \), for \( i \in [1:n] \), \( c_0 = a_0 + \sum_{i=1}^{n} a_i r_i \), and the activation function \( h = g_1 \). Then the FAR’s output \( f \) satisfies \( f = o \), where \( o \) is given by (3.1).

This result provides a useful tool for KBD of feedforward ANNs. The next example demonstrates this.

### 3.1.2 Example 2: Knowledge-Based Design of a Feedforward ANN

Consider the problem of designing an ANN with two binary inputs \( z_1, z_2 \in \{0, 1\} \), and a single output \( f = \text{not}(\text{xor} \ z_1 \ z_2) \). In other words, the ANN should compute the xornot function.

Suppose that our initial knowledge is the truth table of the function xornot\((z_1, z_2)\), shown graphically in Fig. 3.3. It is easy to see that the two input combinations for which \( \text{xornot}(z_1, z_2) = 1 \) (denoted by \( \times \)) are inside the region bounded by the two parallel lines \( z_2 - z_1 = 1/2 \) and \( z_2 - z_1 = -1/2 \). Hence, letting \( p := z_2 - z_1 \), we can state the required operation in symbolic form as:

If \( -1/2 < p < 1/2 \) then \( f = 1 \); otherwise \( f = 0 \).

Motivated by (2.4), we restate this using the following set of rules:

- **R** \(_1\): If \( p \) is smaller than \( 1/2 \) and \( p \) is larger than \(-1/2 \), Then \( f = 1 \),
- **R** \(_2\): If \( p \) is smaller than \( 1/2 \) and \( p \) is smaller than \(-1/2 \), Then \( f = 0 \),
- **R** \(_3\) : If \( p \) is larger than \( 1/2 \) and \( p \) is larger than \(-1/2 \), Then \( f = 0 \).

To transform this FRB into a FAR, we must first find \( a_i, i = 0, 1, 2 \), such that:
Fig. 3.3. The function $\text{xornot}(z_1, z_2)$. $\circ$ denotes zero and $\times$ denotes one. Also shown are the lines $z_2 - z_1 = 1/2$, and $z_2 - z_1 = -1/2$.

\[
a_0 - a_1 + a_2 = 1, \quad a_0 - a_1 - a_2 = 0, \quad \text{and} \quad a_0 + a_1 + a_2 = 0.
\]

This yields
\[
a_0 = 0, \quad a_1 = -1/2, \quad \text{and} \quad a_2 = 1/2.
\] (3.5)

We also need to add the fourth rule:

\begin{itemize}
  \item \textbf{R}_4: If $p$ is \textit{larger than} 1/2 and $p$ is \textit{smaller than} $-1/2$,
  \item Then $f = a_0 + a_1 - a_2 = -1$.
\end{itemize}

Note that the degree-of-firing of this rule will always be very low, suggesting that adding it to the rule-base is harmless.

Suppose that we model the linguistic terms \{\textit{larger than} $k$, \textit{smaller than} $k$\} as in (2.7) with, say, $\alpha = 2$. Then (2.8) yields $\beta(y) = \tanh(y - k)$, so (2.3) holds with:

\[
q = 1, \quad g = \tanh, \quad u = 1, \quad v = k, \quad \text{and} \quad r = 0.
\]

Our four-rule FRB is now a FARBER with $m = 2$ inputs $x_1 = x_2 = p$, $g_i = \tanh$, $u_i = 1$, $v_1 = 1/2$, $v_2 = -1/2$, $r_i = 0$, $q_i = 1$, and the $a_i$s given in (3.5).
Applying Corollary 3.2 shows that the IO mapping \((x_1, x_2) \rightarrow f(x_1, x_2)\) of this FARB is equivalent to the mapping \((y_1, y_2) \rightarrow o(y_1, y_2)\) of the ANN:

\[
o = -\frac{1}{2} \tanh(y_1 - 1/2) + \frac{1}{2} \tanh(y_2 + 1/2),
\]

where \(y_1 = y_2 = p = z_2 - z_1\). It is clear that (3.6) describes an ANN with two neurons in the hidden layer, and a hyperbolic tangent as the activation function.

To make the output binary, we declare the ANN decision to be one if \(o > 1/2\) and zero, otherwise. A contour plot of \(o\) (see Fig. 3.4) shows that \(o\) can indeed be used to compute the xornot function.

Summarizing, we were able to systematically design a suitable ANN by stating our initial knowledge as a FARB, and then using the mathematical equivalence between the FARB and a feedforward ANN. \(\Box\)

The FARB–ANN equivalence can also be used for KBN in RNNs.
3.2 The FARB and First-Order RNNs

Consider a first-order RNN [127] with hidden neurons $s_1, \ldots, s_k$, activation function $h$, input neurons $s_{k+1}, \ldots, s_n$, and weights $w_{ij}$ (see Fig. 3.5). Denoting, for convenience, $y_i(t) := \sum_{j=1}^{n} w_{ij} s_j(t)$, $s_0(t) \equiv 1$, and $w_{i0} = b_i$ yields

$$s_i(t+1) = h(y_i(t) + w_{i0})$$
$$= h\left(\sum_{j=0}^{n} w_{ij} s_j(t)\right),$$

(3.7)

for all $i \in [1 : k]$. We now consider several types of FARBs with an equivalent IO mapping.

3.2.1 First Approach

Consider a two-rule FARB (that is, $m = 1$) with time-invariant parameters $a_0, a_1, q_1$, and $r_1$ satisfying:

$$a_1 q_1 = 1, \quad \text{and} \quad a_0 + r_1 a_1 = 0.$$  

(3.8)

Substituting (3.8) in (2.14) shows that the output of this FARB is
\[ f = g_1(u_1x_1(t) - v_1). \]  

Comparing (3.9) and (3.7) yields the following results.

**Corollary 3.3. (KE from a First-Order RNN)**

Consider the first-order RNN (3.7). Let \( f \) denote the output of a two-rule FARB with: MFs that satisfy (2.3) with \( g_1 = h \), input \( x_1(t) = y_i(t)/u_1 \), parameters that satisfy (3.8), and \( v_1 = -w_{i0} \). Then

\[ f = s_i(t + 1). \]

This result can be used for KE from RNNs. The next example demonstrates this.

**3.2.2 Example 3: Knowledge Extraction from a simple RNN**

Consider the RNN

\[ s_i(t + 1) = \sigma(w_{i1}s_1(t) + w_{i0}). \] (3.10)

Note that this is in the form (3.7) with \( n = 1 \), \( h = \sigma \), and \( y_1 = w_{i1}s_1(t) \).

Corollary 3.3 can be applied to yield a single-input two-rule FARB with an equivalent IO mapping. Suppose that we use the fuzzy terms \{larger than \( k \), smaller than \( k \}\}, modeled using the MFs defined in (2.7), with \( \alpha > 0 \). Then (2.9) implies that

\[ g(z) = \sigma(z), \ u_1 = \alpha, \ v_1 = \alpha k, \ q_1 = 2, \ and \ r_1 = -1, \]

so (3.8) yields

\[ a_0 = a_1 = 1/2. \]

Applying Corollary 3.3 implies that the equivalent FARB is:

- **R**\(_1\): If \( \frac{w_{i1}s_1(t)}{\alpha} \) is larger than \( -\frac{w_{i0}}{\alpha} \), Then \( s_i(t + 1) = 1 \),
- **R**\(_2\): If \( \frac{w_{i1}s_1(t)}{\alpha} \) is smaller than \( -\frac{w_{i0}}{\alpha} \), Then \( s_i(t + 1) = 0 \).
In other words, the IO mapping of this FAR\textsuperscript{B} is identical to the mapping given in (3.10). This provides a \textit{symbolic} representation of the RNN (3.10).

The next result is the converse of Corollary 3.3.

\textbf{Corollary 3.4. (KBD of a First-Order RNN)}

Consider a two-rule FAR\textsuperscript{B} with input $x_1(t)$, output $f(t)$, and parameters satisfying (3.8). Define $y_i(t) = u_1x_1(t), w_{i0} = -v_1$, and the activation function $h = g_1$. Then the FAR\textsuperscript{B}’s output satisfies $f = s_i(t+1)$, where $s_i(t+1)$ is given by (3.7).

Corollaries 3.3 and 3.4 show that the IO mapping of every neuron in a first-order RNN is equivalent to that of a FAR\textsuperscript{B} with two rules. However, a two-rule FAR\textsuperscript{B} is usually too simple to provide useful information on the equivalent RNN. The next section describes an alternative approach.

\textbf{3.2.3 Second Approach}

Let $h^{-1}(\cdot)$ denote the inverse of $h(\cdot)$. We assume that the inverse exists either globally, or at least in some relevant operation domain of the network. We restate (3.7) as follows. For any $i \in [1:k]$:

$$h^{-1}(s_i(t+1)) = \sum_{j=0}^{n} w_{ij}s_j(t) = w_{i0}s_0(t) + \sum_{j=1}^{k} w_{ij}s_j(t) + \sum_{j=k+1}^{n} w_{ij}s_j(t)$$

$$= w_{i0}s_0(t) + \sum_{j=1}^{k} w_{ij}h(\sum_{p=0}^{n} w_{jp}s_p(t-1)) + \sum_{j=k+1}^{n} w_{ij}s_j(t).$$

It will be useful to express this in the form:
\[ h^{-1}(s_i(t + 1)) = w_{i0} h(w_{00} + h^{-1}(s_0(t))) \]
\[ + \sum_{j=1}^{k} w_{ij} h(w_{j0} + \sum_{p=1}^{n} w_{jp} s_p(t - 1)) \]
\[ + \sum_{j=k+1}^{n} w_{ij} h(w_{j0} + h^{-1}(s_j(t))), \]

where we used the fact that \( w_{j0} = 0 \) for \( j = 0 \) and for \( j = [k + 1:n] \).

Letting
\[
\tilde{y}_j(t) := \begin{cases} 
\sum_{p=1}^{n} w_{jp} s_p(t - 1), & \text{if } j \in [1:k], \\
h^{-1}(s_i(t)), & \text{if } j = 0 \text{ or } j \in [k + 1:n],
\end{cases} \tag{3.12}
\]
yields
\[
h^{-1}(s_i(t + 1)) = \sum_{j=0}^{n} w_{ij} h(\tilde{y}_j(t) + w_{j0}), \quad \text{for all } i \in [1:k]. \tag{3.13}
\]

Eq. (3.13) implies that \( h^{-1}(s_i(t + 1)) \) can be represented as the output of a suitable feedforward ANN (see Fig. 3.6).

Comparing (3.13) and (2.14) yields the following results.
Corollary 3.5. (*KE from a First-Order RNN*)
Consider the first-order RNN given by (3.12) and (3.13). Let \( f(t) \) denote the output of a FARB with: MFs that satisfy (2.3) with \( g_i = h \), \( m = n \) inputs \( x_j(t) = \bar{y}_j(t)/u_j \), for \( j \in [1:n] \), and parameters

\[
a_0 = w_{i0} - \sum_{j=1}^{m} w_{ij}r_j/q_j, \quad a_j = w_{ij}/q_j, \quad \text{and} \quad v_j = -w_{j0}, \quad j \in [1:n].
\]

Then \( f(t) = h^{-1}(s_i(t + 1)) \).

Corollary 3.6. (*KBD of a First-Order RNN*)
Consider a FARB with inputs \( x_1(t), \ldots, x_m(t) \) and output \( f(t) \). Define \( n = m, \bar{y}_j(t) = x_j(t)u_j, \) and \( h = g_j, \) for \( j \in [1:n] \), and

\[
w_{i0} = a_0 + \sum_{j=1}^{m} a_j r_j, \quad w_{ij} = a_j q_j, \quad \text{and} \quad w_{j0} = -v_j, \quad j \in [1:n].
\]

Then \( f(t) = h^{-1}(s_i(t + 1)), \) where \( h^{-1}(s_i(t + 1)) \) is given in (3.13).

Another approach for converting the RNN into a FARB or vice-versa is possible when the function \( g \) in (2.3) is piecewise-linear.

### 3.2.4 Third Approach
Assume that each \( g_i \) in (2.3) is a linear function (the results below can be generalized to the case where \( g_i \) is piecewise-linear), that is, there exist \( \bar{u}_i, \bar{v}_i \in \mathbb{R} \) such that

\[
g_i(u_i x_i(t) - v_i) = \bar{u}_i x_i(t) - \bar{v}_i. \tag{3.14}
\]

Then, the FARB output given in (3.2) can be restated as

\[
f(t) = a_0 + \sum_{i=1}^{m} a_i r_i + \sum_{i=1}^{m} a_i q_i (\bar{u}_i x_i(t) - \bar{v}_i)
\]

\[
= a_0 + \sum_{i=1}^{m} a_i (r_i - q_i \bar{v}_i) + \sum_{i=1}^{m} a_i q_i \bar{u}_i x_i(t). \tag{3.15}
\]
On the other-hand, recall that the first-order RNN (3.7) can be described by (3.11), that is,

$$h^{-1}(s_i(t + 1)) = \sum_{j=0}^{n} w_{ij} s_j(t).$$  \hspace{1cm} (3.16)

Comparing (3.15) to (3.16) yields the following result.

**Corollary 3.7. (KE from a First-Order RNN)**

Consider the first-order RNN given by (3.16). Let $f(t)$ denote the output of a FARB with: MFs that satisfy (2.3) with $g_i$ satisfying (3.14), $m = n$ inputs $x_j(t) = s_j(t)/\tilde{u}_j$, for $j \in [1:n]$, and parameters

$$a_0 = w_{i0} - \sum_{j=1}^{m} w_{ij}(r_j - q_j\tilde{v}_j)/q_j, \text{ and } a_j = w_{ij}/q_j, \text{ for } j \in [1:n].$$

Then $f(t) = h^{-1}(s_i(t + 1))$.

This result provides a useful mechanism for KE from RNNs. The next example demonstrates this.

### 3.2.5 Example 4: Knowledge Extraction from an RNN

Consider the one-input-one-output RNN depicted in Fig. 3.7. Assume that the initial condition is $s(1) = 1$. The RNN is then described by:

$$s(t + 1) = \sigma_L(s(t) + I(t) - 1), \quad s(1) = 1. \quad (3.17)$$

We consider the case where the input is binary: $I(t) \in \{0, 1\}$ for all $t$. 

---

Fig. 3.7. A simple RNN.
We now transform this RNN into an equivalent FARB. It will be convenient to rewrite (3.17) in the form

\[ s_1(t + 1) = \sigma_L(s_1(t) + s_2(t) - 1), \]  

(3.18)

with \( s_1(t) := s(t), \) and \( s_2(t) := I(t). \) Note that this is a special case of (3.16) with: \( k = 1, \) \( n = 2, \) \( h = \sigma_L, \) and the weights:

\[ w_{10} = -1, \quad w_{11} = 1, \quad w_{12} = 1. \]  

(3.19)

Suppose that we characterize all the variables in the equivalent FARB using the fuzzy terms \{equals 1, equals 0\}, modeled using the MFs:

\[ \mu_-(u) := \sigma_L(u) \quad \text{and} \quad \mu_0(u) := \sigma_L(1 - u). \]  

(3.20)

The definition of \( \beta(\cdot) \) yields

\[
\beta(z) = \frac{\sigma_L(z) - \sigma_L(1 - z)}{\sigma_L(z) + \sigma_L(1 - z)}
= \sigma_L(z) - \sigma_L(1 - z).
\]

Note that \( \sigma_L(z) = z \) for any \( z \in [0, 1]. \) Thus, in this linear range:

\[
\beta(z) = 2z - 1
= 2\sigma_L(z) - 1,
\]

so (2.3) and (3.14) hold with:

\[ g = \sigma_L, \quad u = 1, \quad v = 0, \quad q = 2, \quad r = -1, \quad \tilde{u} = u = 1, \quad \text{and} \quad \tilde{v} = v = 0. \]  

(3.21)

Corollary 3.7 now implies that the RNN (3.18) is equivalent to a four-rule FARB with \( m = 2 \) inputs \( x_1(t) = s_1(t)/\tilde{u}_1 = s(t), \) \( x_2(t) = s_2(t)/\tilde{u}_2 = I(t). \) and parameters
3.2 The FARB and First-Order RNNs

\[ a_1 = w_{11}/q_1, \]
\[ a_2 = w_{12}/q_2, \]
\[ a_0 = w_{10} - (w_{11}(r_1 - q_1 \tilde{v}_1)/q_1 + w_{12}(r_2 - q_2 \tilde{v}_2)/q_2). \]

Using (3.19) and (3.21) yields

\[ a_0 = 0, \quad a_1 = a_2 = 1/2. \]

Summarizing, the equivalent FARB is

- **R1**: If \( s(t) \) equals 1 and \( I(t) \) equals 1, Then \( \sigma^{-1}_L(s(t + 1)) = 1, \)
- **R2**: If \( s(t) \) equals 1 and \( I(t) \) equals 0, Then \( \sigma^{-1}_L(s(t + 1)) = 0, \)
- **R3**: If \( s(t) \) equals 0 and \( I(t) \) equals 1, Then \( \sigma^{-1}_L(s(t + 1)) = 0, \)
- **R4**: If \( s(t) \) equals 0 and \( I(t) \) equals 0, Then \( \sigma^{-1}_L(s(t + 1)) = -1. \)

This provides a symbolic representation of the RNN (3.17). This FARB can be further simplified as follows. Since \( \sigma_L(-1) = \sigma_L(0) = 0, \) and since rule \( R_1 \) is the only rule where the Then-part satisfies \( \sigma_L^{-1}(s(t + 1)) > 0, \) the FARB can be summarized as:

- If \( s(t) \) equals 1 and \( I(t) \) equals 1, Then \( s(t + 1) = \sigma_L(1) = 1; \)
- Else \( s(t + 1) = 0. \)

This can be stated as:

- If \( I(t) \) equals 1, Then \( s(t + 1) = s(t); \)
- Else \( s(t + 1) = 0. \)

Using this description, it is straightforward to understand the RNN functioning. Recall that \( s(t) \) is initialized to 1. It follows that \( s(t) \) will remain 1 until the first time the input is \( I(t) = 0. \) Once this happens, \( s(t + 1) \) is set to zero, and will remain 0 from thereon. In other words, the RNN output is:

\[ s(t + 1) = \begin{cases} 1, & \text{if } I(\tau) = 1 \text{ for all } \tau \leq t, \\ 0, & \text{otherwise.} \end{cases} \]
If we regard the RNN as a formal language recognizer (see Section 5.3.1 below), with $s(t + 1) = 1$ interpreted as Accept, and $s(t + 1) = 0$ as Reject, then the RNN accepts a binary string if it does not include any zero bits. Summarizing, the transformation into a FAR is provides a comprehensible explanation of the RNN functioning.

By comparing (3.15) to (3.16), it is also possible to derive the following result which is the converse of Corollary 3.7.

**Corollary 3.8. (KBD of a First-Order RNN)**

Consider a FAR with inputs $x_1(t), \ldots, x_m(t)$, output $f(t)$ and functions $g_i$ satisfying (3.14). Define $n = m$, $s_j(t) = x_j(t)u_j$ for $j \in [1:n]$, and

$$w_{i0} = a_0 + \sum_{j=1}^{m} a_j(r_j - q_jv_j), \quad \text{and} \quad w_{ij} = a_jq_j, \quad j \in [1:n].$$

Then

$$f(t) = h^{-1}(s_i(t + 1)),$$

where $h^{-1}(s_i(t + 1))$ is given in (3.16).

This result can be used for KBD of RNNs. The next example demonstrates this.

**3.2.6 Example 5: Knowledge-Based Design of an RNN**

Consider the following problem. Design an RNN that accepts a binary string $I(1), I(2), \ldots$ as an input. If the input string contained the bit 1, the RNN’s output should be $s(t + 1) = 0$. Otherwise, that is, if $I(1) = I(2) = \cdots = I(t) = 0$, the RNN’s output should be $s(t + 1) = 1$.

Our design is based on stating the required functioning as a FAR and then using Corollary 3.8 to obtain the equivalent RNN. The design concept is simple. We initialize $s(1) = 1$. If $I(t) = 1$, then we set $s(t + 1) = 0$; if $I(t) = 0$, then $s$ is unchanged. Thus, once a 1 appears in the input, $s(\cdot)$ will be set to zero, and remain zero from thereon.
3.2 The FARB and First-Order RNNs

Fig. 3.8. Graphical representation of the RNN described by Eq. (3.22).

We can state the desired behavior of \( s(t) \) in the following form:

\[ R_1: \text{If } I(t) \text{ equals 1, Then } s(t+1) = 0, \]
\[ R_2: \text{If } I(t) \text{ equals 0, Then } s(t+1) = s(t). \]

This is a two-rule FARB with a single input \( x_1(t) = I(t) \), and

\[ a_0(t) = s(t)/2, \]
\[ a_1(t) = -s(t)/2. \]

We model the fuzzy terms \{\text{equals 1}, \text{equals 0}\} as in (3.20), so (2.3) and (3.14) hold with:

\[ g_1 = \sigma_L, \ u_1 = 1, \ v_1 = 0, \ q_1 = 2, \ r_1 = -1, \ \bar{u}_1 = u_1 = 1, \ \bar{v}_1 = v_1 = 0. \]

Applying Corollary 3.8 yields \( n = 1, \ s_1(t) = I(t), \ w_{10} = s(t), \) and \( w_{11} = -s(t). \) Substituting these parameters in (3.16) yields

\[ \sigma_L^{-1}(s(t+1)) = s(t) - I(t)s(t), \]

or

\[ s(t+1) = \sigma_L(s(t) - I(t)s(t)). \quad (3.22) \]

Clearly, this describes the dynamics of an RNN with a single neuron \( s(t) \) and a single input \( I(t) \) (see Fig. 3.8). Recalling that \( s(1) = 1 \), it is easy to verify that \( s(t+1) = 0 \) iff there exists \( j \in [1:t] \) such that \( I(j) = 1 \). Otherwise, \( s(t+1) = 1 \). Thus, the designed RNN indeed solves the given problem.

The next section describes a connection between the FARB and another type of RNN.
3.3 The FARB and Second-Order RNNs

Second-order RNNs, introduced by Pollack in 1987 [134], are a generalization of first-order RNNs, and, in particular, can solve problems that first-order RNNs cannot [53]. In second-order RNNs, the connection weights are linear functions of the neurons values [135]:

\[ w_{ij}(t) = \sum_{l=0}^{n} w_{ijkl}s_l(t). \]

Denote the hidden neurons by \( s_1, \ldots, s_k \), the bias neuron by \( s_0(t) \equiv 1 \), and the input neurons by \( s_{k+1}, \ldots, s_n \), then the dynamics of a second-order RNN is given by

\[ s_i(t+1) = h(\sum_{j=0}^{n} w_{ijl}s_l(t)s_j(t)) = h(\sum_{j=0}^{n} \sum_{l=0}^{n} w_{ijkl}s_l(t)s_j(t)), \quad i \in [1:k], \tag{3.23} \]

where \( h \) is the activation function. The parameter \( w_{ij0} \) is the weight of the connection from neuron \( j \) to neuron \( i \), and \( w_{i00} \) is the bias of neuron \( i \). Note that, by definition,

\[ w_{ij0} = w_{i0j}. \tag{3.24} \]

Eq. (3.23) yields

\[ h^{-1}(s_i(t+1)) = \sum_{j=0}^{n} \sum_{l=0}^{n} w_{ijl}s_l(t)s_j(t) = w_{i00} + 2 \sum_{j=1}^{n} w_{ij0}s_j(t) + \sum_{j=1}^{n} \sum_{l=1}^{n} w_{ijkl}s_l(t)s_j(t), \tag{3.25} \]

where the last equation follows from (3.24).

To determine a FARB with an equivalent IO mapping, consider the case where the FARB parameters satisfy:
\[ a_k(t) = d_k + \sum_{j=1}^{m} d_{kj}x_j(t), \quad k \in [0:m]. \quad (3.26) \]

For the sake of convenience, denote \( x_0(t) \equiv 1 \) and \( d_{k0} = d_k \), so \( a_k(t) = \sum_{j=0}^{m} d_{kj}x_j(t) \). Then, (2.14) yields

\[ f = \sum_{j=0}^{m} d_{0j}x_j(t) + \sum_{k=1}^{m} d_{kj}r_kx_j(t) \\
+ \sum_{k=1}^{m} \sum_{j=0}^{m} d_{kj}q_kx_j(t)g_k(u_kx_k(t) - v_k). \]

Assume also that each \( g_k \) is a linear function so that (3.14) holds. Then

\[ f = d_{00} + \sum_{k=1}^{m} d_{k0}(r_k - q_k\bar{v}_k) \\
+ \sum_{j=1}^{m} \left( d_{0j} + d_{j0}q_j\bar{u}_j + \sum_{k=1}^{m} d_{kj}(r_k - q_k\bar{v}_k) \right) x_j(t) \\
+ \sum_{k=1}^{m} \sum_{l=1}^{m} d_{kl}q_k\bar{u}_kx_k(t)x_l(t). \quad (3.27) \]

Comparing (3.27) and (3.25) yields the following results.

**Corollary 3.9. (KE from a Second-Order RNN)**

Consider the second-order RNN (3.25). Let \( f \) denote the output of a FARB with: \( m = n \) inputs \( x_j(t) = s_j(t) \), MFs that satisfy (2.3) and (3.14) and parameters that satisfy (3.26) and

\[ d_{kl} = w_{ikl}/(q_k\bar{u}_k), \]

\[ d_{0l} + d_{l0}q_l\bar{u}_l + \sum_{k=1}^{m} \frac{w_{ikl}}{q_k\bar{u}_k}(r_k - q_k\bar{v}_k) = 2w_{l0}, \]

\[ d_{00} + \sum_{k=1}^{m} d_{k0}(r_k - q_k\bar{v}_k) = w_{00}, \]
for \( k, l \in [1:m] \). Then \( f = h^{-1}(s_i(t + 1)) \).

In other words, we can transform the dynamics of every neuron \( s_i \) in the second-order RNN into an equivalent FARB.

The next result is the converse of Corollary 3.9, namely, it provides a transformation from a FARB into an equivalent second-order RNN.

**Corollary 3.10. (KBD of a Second-Order RNN)**
Consider a FARB with inputs \( x_1(t), \ldots, x_m(t) \), output \( f(t) \), MFs such that (3.14) holds, and parameters that satisfy (3.26). Define \( n = m \), \( h = g_i \), \( s_j(t) = x_j(t) \) for \( j \in [1:n] \), and weights

\[
\begin{align*}
    w_{ikl} &= d_{kl}q_k\tilde{u}_k, \\
    w_{i0l} &= (d_{0l} + d_{l0}q_l\tilde{u}_l + \sum_{k=1}^m d_{kl}(r_k - q_k\tilde{v}_k))/2, \\
    w_{i00} &= d_{00} + \sum_{k=1}^m d_{k0}(r_k - q_k\tilde{v}_k),
\end{align*}
\]

for \( k, l \in [1:n] \). Then \( f(t) = h^{-1}(s_i(t + 1)) \), where \( h^{-1}(s_i(t + 1)) \) is given in (3.25).

**3.4 Summary**

In this chapter we studied in detail the mathematical equivalence between the FARB and various types of ANNs. We showed that the FARB–ANN equivalence holds for a large variety of ANNs, regardless of their specific architecture and parameter values. Since the FARB is a standard FRB, this enables the application of tools from the theory of ANNs to FARBs, and vice versa.

Given an ANN, we can immediately determine a suitable FARB with the same IO mapping, and thus provide a a symbolic description of the ANN functioning.

Conversely, consider the problem of designing an ANN for solving a given problem. In many cases, some initial knowledge about the
problem domain is known. Designing a symbolic FARB based on this knowledge yields an IO mapping that can be immediately realized as a suitable ANN.

These ideas were demonstrated using simple examples. Applications to larger-scale problems are studied in Chapters 5 and 6 below.