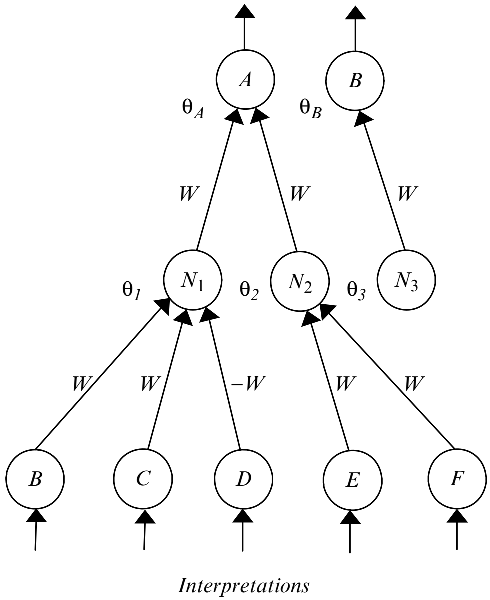

Fig. 4.3 Sketch of a neural network for the logic program $\mathcal{P}$ in Example 1

output neuron B should feed input neuron B such that $\mathcal{N}$ is used to iterate $T_{\mathcal{P}}$, the fixed-point operator[2] of $\mathcal{P}$. $\mathcal{N}$ will eventually converge to a stable state which is identical to the stable model of $\mathcal{P}$.

Notation Given a general logic program $\mathcal{P}$, let:

q denote the number of clauses r_l ($1 \leq l \leq q$) occurring in $\mathcal{P}$;
υ denote the number of literals occurring in $\mathcal{P}$;
A_{min} denote the minimum activation for a neuron to be *active* (or, analogously, for its associated literal to be assigned a truth value $true$), $A_{min} \in (0,1)$;
A_{max} denote the maximum activation when a neuron is not active (or when its associated literal is $false$), $A_{max} \in (-1,0)$;
$h(x) = 2/(1+e^{-\beta x}) - 1$, the bipolar semilinear activation function;[3]
$g(x) = x$, the standard linear activation function;
$s(x) = y$, the standard nonlinear activation function ($y = 1$ if $x > 0$, and $y = 0$ otherwise), also known as the step function;
W and $-W$ denote the weights of connections associated with positive and negative literals, respectively;
θ_l denote the threshold of the hidden neuron N_l associated with clause r_l;
θ_A denote the threshold of the output neuron A, where A is the head of clause r_l;
k_l denote the number of literals in the body of clause r_l;

[2] Recall that the mapping $T_{\mathcal{P}}$ is defined as follows. Let I be a Herbrand interpretation; then $T_{\mathcal{P}}(I) = \{A_0 \mid L_1, \ldots, L_n \rightarrow A_0$ is a ground clause in $\mathcal{P}$ and $\{L_1, \ldots, L_n\} \subseteq I\}$.

[3] We use the bipolar semilinear activation function for convenience (see Sect. 3.1). Any monotonically increasing activation function could have been used here.

p_l denote the number of positive literals in the body of clause r_l;
n_l denote the number of negative literals in the body of clause r_l;
μ_l denote the number of clauses in $\mathcal{P}$ with the same atom in the head, for each clause r_l;
$MAX_{r_l}(k_l, \mu_l)$ denote the greater element of k_l and μ_l for clause r_l; and
$MAX_{\mathcal{P}}(k_1, \ldots, k_q, \mu_1, \ldots, \mu_q)$ denote the greatest element of all k's and μ's of $\mathcal{P}$.

We also use $\overrightarrow{k}$ as a shorthand for $(k_1, \ldots, k_q)$, and $\overrightarrow{\mu}$ as a shorthand for $(\mu_1, \ldots, \mu_q)$.

For instance, for the program $\mathcal{P}$ of Example 1, $q = 3$, $\upsilon = 6$, $k_1 = 3$, $k_2 = 2$, $k_3 = 0$, $p_1 = 2$, $p_2 = 2$, $p_3 = 0$, $n_1 = 1$, $n_2 = 0$, $n_3 = 0$, $\mu_1 = 2$, $\mu_2 = 2$, $\mu_3 = 1$, $MAX_{r_1}(k_1, \mu_1) = 3$, $MAX_{r_2}(k_2, \mu_2) = 2$, $MAX_{r_3}(k_3, \mu_3) = 1$, and $MAX_{\mathcal{P}}(k_1, k_2, k_3, \mu_1, \mu_2, \mu_3) = 3$.

In the Translation Algorithm below, we define A_{min}, W, θ_l, and θ_A such that the conditions (**C1**) and (**C2**) above are satisfied. Equations 4.1, 4.2, 4.3, and 4.4 below are obtained from the proof of Theorem 8 [80]. We assume, for mathematical convenience and without loss of generality, that $A_{max} = -A_{min}$. In this way, we associate the truth value *true* with values in the interval $(A_{min}, 1)$, and the truth value *false* with values in the interval $(-1, -A_{min})$.

Theorem 8 guarantees that values in the interval $[-A_{min}, A_{min}]$ do not occur in the network with weights W and thresholds θ, but, informally, this interval may be associated with a third truth value *unknown*[4]. The proof of Theorem 8 presented in [66] is reproduced here for the sake of completeness, as several proofs presented in the book will make reference to it.

We start by calculating $MAX_{\mathcal{P}}(\overrightarrow{k}, \overrightarrow{\mu})$ such that

$$A_{min} > \frac{MAX_{\mathcal{P}}(\overrightarrow{k}, \overrightarrow{\mu}) - 1}{MAX_{\mathcal{P}}(\overrightarrow{k}, \overrightarrow{\mu}) + 1}. \tag{4.1}$$

CILP Translation Algorithm

1. Calculate the value of W such that the following is satisfied:

$$W \geq \frac{2}{\beta} \cdot \frac{\ln(1 + A_{min}) - \ln(1 - A_{min})}{MAX_{\mathcal{P}}(\overrightarrow{k}, \overrightarrow{\mu})(A_{min} - 1) + A_{min} + 1}. \tag{4.2}$$

2. For each clause r_l of $\mathcal{P}$ of the form $L_1, \ldots, L_k \rightarrow A$ $(k \geq 0)$:

   (a) Create input neurons $L_1, \ldots, L_k$ and an output neuron A in $\mathcal{N}$ (if they do not exist yet).
   (b) Add a neuron N_l to the hidden layer of $\mathcal{N}$.

[4] If a network obtained by the Translation Algorithm is then trained by examples with the use of a learning algorithm that does not impose any constraints on the weights, values in the interval $[-A_{min}, A_{min}]$ may occur and should be interpreted as unknown by following a three-valued interpretation.

(c) Connect each neuron L_i $(1 \leq i \leq k)$ in the input layer to the neuron N_l in the hidden layer. If L_i is a positive literal, then set the connection weight to W; otherwise, set the connection weight to $-W$.
(d) Connect the neuron N_l in the hidden layer to the neuron A in the output layer and set the connection weight to W.
(e) Define the threshold (θ_l) of the neuron N_l in the hidden layer as

$$\theta_l = \frac{(1+A_{min})\,(k_l-1)}{2}W. \tag{4.3}$$

a. Define the threshold (θ_A) of the neuron A in the output layer as

$$\theta_A = \frac{(1+A_{min})\,(1-\mu_l)}{2}W. \tag{4.4}$$

3. Set $g(x)$ as the activation function of the neurons in the input layer of $\mathcal{N}$. In this way, the activation of the neurons in the input layer of $\mathcal{N}$ given by each input vector $\mathbf{i}$ will represent an interpretation for $\mathcal{P}$.
4. Set $h(x)$ as the activation function of the neurons in the hidden and output layers of $\mathcal{N}$. In this way, a gradient descent learning algorithm, such as backpropagation, can be applied to $\mathcal{N}$.
5. If $\mathcal{N}$ needs to be fully connected, set all other connections to zero.

Theorem 8. *[66] For each propositional general logic program $\mathcal{P}$, there exists a feedforward artificial neural network $\mathcal{N}$ with exactly one hidden layer and semilinear neurons such that $\mathcal{N}$ computes the fixed-point operator $T_{\mathcal{P}}$ of $\mathcal{P}$.*

Proof. $(\leftarrow)$ '$A \geq A_{min}$ if $L_1,\ldots,L_k$ is satisfied by $\mathbf{i}$'. Assume that the p_l positive literals in $L_1,\ldots,L_k$ are $true$, and the n_l negative literals in $L_1,\ldots,L_k$ are $false$. Consider the mapping from the input layer to the hidden layer of $\mathcal{N}$. The input potential (I_l) of N_l is minimum when all the neurons associated with a positive literal in $L_1,\ldots,L_k$ are at A_{min}, while all the neurons associated with a negative literal in $L_1,\ldots,L_k$ are at $-A_{min}$. Thus, $I_l \geq p_l A_{min} W + n_l A_{min} W - \theta_l$ and, assuming $\theta_l = ((1+A_{min})\,(k_l-1)/2)W$, $I_l \geq p_l A_{min} W + n_l A_{min} W - ((1+A_{min})\,(k_l-1)/2)W$.

If $h(I_l) \geq A_{min}$, i.e. $I_l \geq -\frac{1}{\beta} ln((1-A_{min})/(1+A_{min}))$, then N_l is active. Therefore, Equation 4.5 must be satisfied:[5]

$$p_l A_{min} W + n_l A_{min} W - \frac{(1+A_{min})\,(k_l-1)}{2}W \geq -\frac{1}{\beta}\ln\left(\frac{1-A_{min}}{1+A_{min}}\right). \tag{4.5}$$

Solving Equation 4.5 for the connection weight (W) yields Equations 4.6 and 4.7, given that $W > 0$:

[5] Throughout, we use the word 'Equation', as in 'Equation 4.5', even though 'Equation 4.5' is an inequality.