MODELING DEDUCTION WITH RECURRENT NEURAL NETWORKS

Arnošt Veselý[*]

Abstract: In the paper, we focus on reasoning with IF-THEN rules in propositional fragment of predicate calculus and on its modeling with neural networks. At first, IF-THEN deduction from facts is defined. Then it is proved that for any non-contradictory set of IF-THEN rules and literals (representing facts) there exists a layered recurrent network with 2 hidden layers that can specify all IF-THEN deducible literals. If we denote the set of all literal IF-THEN consequences as $\boldsymbol{D}_0$ and the set of all literal logical consequences as $\boldsymbol{D}$, then obviously $\boldsymbol{D}_0 \subset \boldsymbol{D}$. Thus, $\boldsymbol{D}_0$ can be considered to be an approximation of $\boldsymbol{D}$. Using the designed network for simulation of contradiction proof, the approximation $\boldsymbol{D}_0$ may be easily refined. Furthermore, the network may also be used for determination of $\boldsymbol{D}$. However, the algorithm that realizes necessary network computations has exponential complexity.

Key words: *IF-THEN rules, logical deduction, neural networks*

Received: June 28, 2011
Revised and accepted: April 16, 2012

1. Introduction

Today's models of expert decision-making used in the field of Cognitive Science suppose that expert knowledge about observed or controlled system can be decomposed into small pieces or chunks, which can be expressed as IF-THEN rules [1], [2]. The current state of the system is often described by a set of simple facts having the form of literals (literals are predicates or their negatives). Thus, an important part of expert decision-making is the search for other valid facts that are not known but that must be valid due to the IF-THEN rules.

The seminal work exploring neural network modeling of deduction with IF-THEN rules is the article of Towell and Shavlik [3]. The authors assumed general knowledge to be structured in a set of IF-THEN rules interpreted as logical implications. Knowledge based on observation was represented with a set of observation literals with known truth-values. The logical deduction was represented by means

[*]Arnošt Veselý
Czech University of Life Sciences, Kamycka 129, Prague, Czech Republic, +420 22438 2215, E-mail: vesely@pef.czu.cz

of oriented graph with a tree structure. Tree leaves represented observation literals. The non-leave nodes represented deduced literals. The graph topology was determined by IF-THEN rules contained in the knowledge database. Deduction tree was modeled with a layered neural network. On the network input truth-values of observation literals were put and the network carried out its computation. After computation the output values of neurons in the output layer were interpreted as deduced truth-values of the literals assigned to the output neurons.

Results relevant to deduction modeling with neural networks were acquired also in the field of logical programming. A logical program in propositional logic is a finite set of Horn clauses of the form $A \leftarrow L_1 \wedge \cdots \wedge L_n$, where A is a predicate and L_i are literals. The case n=0 is also allowed. It is called unit clause or fact and denoted $A \leftarrow$. In [4] and [5] a method for constructing a layered neural network that computes consequence operation of arbitrary logical program was described. If we confined ourselves only to those IF-THEN rules that can be equivalently written as Horn clauses, then this neural network might be used for IF-THEN deduction modeling.

Stenning [6] proposed to model deduction with IF-THEN rules that can be interpreted as Horn clauses by means of a recurrent neural network with a special and unusual architecture. The model was based on three-valued logic. Truth-values **t** (true), **f** (false) and **u** (undecided) were coded with binary couples **t** = (1, 0), **f** = (0, 1) and **u** = (0, 0). The network consisted of two parallel layers U^+ and U^-. The layer U^+ was a layered recurrent network that computed the first component of the truth-value. The layer U^- had the same architecture as layer U^+ and it computed the second component of the truth-value. A couple of neurons that occupied the corresponding locations in U^+ and U^- layers represented one predicate. Both neurons of the couple were connected together and inhibited each other.

The drawback of the method of Towel and Shavlik [3] is that the architecture of their network depends on the set of observation literals with known truth-values. If truth-values of other observation literals are at our disposal, the network architecture is different. The solutions given in [4] and [5] use fixed architecture. Nevertheless, they restrict the set of IF-THEN rules only to the set of IF-THEN rules equivalent to Horn clauses. The same restriction is postulated in the solution of Stenning [6]. Moreover, Stenning's network has nonstandard and complicated architecture.

In the paper, we propose a solution that escapes drawbacks mentioned above. The proposed network has a fixed, simple layered recurrent architecture and can be used for arbitrary set of IF-THEN rules. The structure of the paper is the following. In Section 2 we are giving the definition of IF-THEN deduction of a literal L from facts $\boldsymbol{F}$ using set of IF-THEN rules $\boldsymbol{R}$. If a literal L can be IF-THEN deduced from $\boldsymbol{F}$ using $\boldsymbol{R}$, we write $\boldsymbol{R}$, $\boldsymbol{F} \mapsto L$. Obviously, if $\boldsymbol{R}$, $\boldsymbol{F} \mapsto L$, then L is logical consequence of $\boldsymbol{R}$ and $\boldsymbol{F}$, i.e. $\boldsymbol{R}$, $\boldsymbol{F} \vdash L$ must hold. Lemma 1 shows that the inverse implication does not hold. Therefore, if an expert applies only IF-THEN rules to get consequences (more precisely, if he uses only IF-THEN deduction defined in Section 2), he cannot, in general, get all logically deducible facts. However, IF-THEN deduction is quick and can be carried out with quite a simple recurrent neural network consisting of standard neurons. How such a

network can be created for a given set of IF-THEN rules $\boldsymbol{R}$ is described in Section 3. In Section 4 we prove that for a given set of facts $\boldsymbol{F}$ the network could quickly compute all its literal IF-THEN consequences L. At the end of the section some useful properties of network computation procedure are proved in Lemmas 2 and 3. Section 5 contains conclusion. Here we outline how results of Lemmas 2 and 3 might be used in the search for all literal logical consequences. If we denote the set of all literal IF-THEN consequences as $\boldsymbol{D}_0$ and the set of all literal logical consequences as $\boldsymbol{D}$, then obviously $\boldsymbol{D}_0 \subset \boldsymbol{D}$. Thus, $\boldsymbol{D}_0$ can be considered to be an approximation of $\boldsymbol{D}$. Lemma 2 provides the possibility to use the network for a more precise and quick approximation of $\boldsymbol{D}$. Quick here means that the algorithm that realizes the necessary network computations has linear complexity. The Lemma 3 presents a property of network computation that enables to use the network also for determination of all literal logical consequences $\boldsymbol{D}$. However, the complexity of the resulting algorithm is exponential.

2. Deduction with IF-THEN Rules

In this paper we consider deduction of consequences from a set of IF-THEN rules $\boldsymbol{R}$ and a set of facts $\boldsymbol{F}$. The formulas consist of ground predicates P_1, P_2, ... and the following logical operators: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) and $\supset$ (implication). Predicates are also called atoms. Literals are predicates or their negations. Predicates are also called positive literals and negations of predicates negative literals. IF-THEN rules are formulas $L_1 \wedge \cdots \wedge L_n \supset L$, where $L_1, \ldots, L_n$, L are literals. A disjunction of literals is called clause. The void clause is denoted $\square$. Facts are literals. If a formula φ is a logical consequence of formulas from $\boldsymbol{R} \cup \boldsymbol{F}$, we write $\boldsymbol{R}, \boldsymbol{F} \vdash \varphi$. An assignment of truth-values 0 (false) or 1 (true) to predicates is called interpretation. If $\boldsymbol{I}$ is interpretation, then $\boldsymbol{I}(\varphi)$ denotes the truth-value of φ in the interpretation $\boldsymbol{I}$. An interpretation in which all formulas $\varphi \in \Phi$ are valid (have truth-value 1) is called model of Φ. A set of formulas that has a model is called non-contradictory.

In the paper we will explore IF-THEN deduction, which is a special type of logical deduction.

Definition 1 Assume that $\boldsymbol{R} \cup \boldsymbol{F}$ is a non-empty and non-contradictory set of IF-THEN rules and facts, the elements of which contain predicates $P_1, \ldots, P_n$. A sequence of formulas $\boldsymbol{D} = (A_1, \ldots, A_p)$ with the following properties 1) and 2) is called IF-THEN deduction of a literal L from $\boldsymbol{R} \cup \boldsymbol{F}$.

1) Each A_i, $1 \leq i \leq p$ is a fact from $\boldsymbol{F}$ or it is a result of application of a derivation rule

$$\frac{A_{j1}, \ldots, A_{jn},\ A_{j1} \wedge \cdots \wedge A_{jn} \supset A_i}{A_i},$$

where literals $A_{j1}, \ldots, A_{jn}$ precede A_i in $\boldsymbol{D}$, and

$$(A_{j1} \wedge \cdots \wedge A_{jn} \supset A_i) \in \boldsymbol{R}.$$

2) $A_p = L$

If there exists IF-THEN deduction of L from $\boldsymbol{R}\cup\boldsymbol{F}$, we write $\boldsymbol{R}$, $\boldsymbol{F}\mapsto L$ and we say that L is IF-THEN consequence of $\boldsymbol{R}\cup\boldsymbol{F}$.

Lemma 1 If $\boldsymbol{R}$, $\boldsymbol{F}\mapsto L$, then $\boldsymbol{R}$, $\boldsymbol{F}\vdash L$. However, the statement does not hold conversely.

Proof Obviously $\boldsymbol{R}$, $\boldsymbol{F}\vdash L$ follows from $\boldsymbol{R}$, $\boldsymbol{F}\mapsto L$. Converse implication does not hold as the following example demonstrates. Assume $\boldsymbol{R}=\{P_1 \supset P_2, \neg P_2 \supset P_1\}$, $\boldsymbol{F}=\emptyset$. Formula $P_1 \supset P_2$ is equivalent to the clause $\neg P_1 \vee P_2$ and formula $\neg P_2 \supset P_1$ is equivalent to the clause $P_1 \vee P_2$. Using resolution principle (see for example [7]) void clause $\square$ can be easily derived from the set of formulas $\{\neg P_1 \vee P_2, P_1 \vee P_2, \neg P_2\}$ (see Fig. 1). Therefore, the set of formulas $\boldsymbol{R}\cup\boldsymbol{F}\cup\{\neg P_2\}$ is contradictory and $\boldsymbol{R}$, $\boldsymbol{F}\vdash P_2$ must hold. On the other hand, from the Definition 1 we see that IF-THEN deduction of P_2 from $\boldsymbol{R}\cup\boldsymbol{F}$ cannot exist.

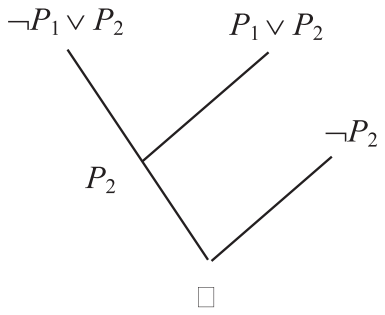

Fig. 1 *Derivation of the void clause* $\square$ *from* $\neg P_1 \vee P_2$, $P_1 \vee P_2$ *and* $\neg P_2$ *using resolution principle.*

3. Neural Network for Modeling Deduction

We will show that IF-THEN deduction from a set of IF-THEN rules $\boldsymbol{R}$ and a set of facts $\boldsymbol{F}$ can be realized with a recurrent neural network. The network will consist of neurons and receptors described below.

Neuron is a threshold element (see Fig. 2) with threshold $\theta \geq 0$ and the output function $\mu(x)$, where $\mu(x)$=1 if $x \geq \nu$, $\mu(x)$=1/2 if $-\nu \leq x < \nu$ and $\mu(x)$=0 if $x < -\nu$. The parameter $\nu \geq 0$ is a non-negative real number. For input vector $\mathbf{x}$, weight vector $\mathbf{w}$ and threshold θ the neuron output y is

$$y = \mu(\mathbf{w}^T\mathbf{x} - \theta).$$

Here the vectors $\mathbf{x}$ and $\mathbf{w}$ are column vectors and $\mathbf{w}^T$ is the row vector obtained by the transposition of the vector $\mathbf{w}$. The expression ($\mathbf{w}^T\mathbf{x}$-θ) represents postsynaptic potential. We will also call it simply neuron input. Neuron with parameters θ and ν we will denote $N_{\theta,\nu}$.

Receptors are so called "degenerative" neurons used for introducing network input. Receptor output value equals its input value.

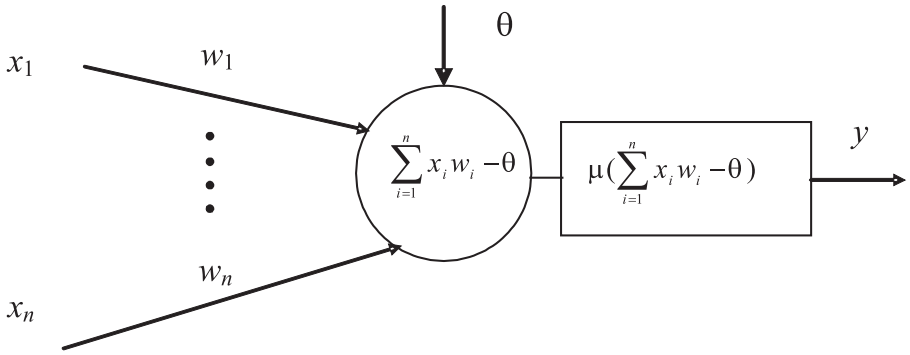

Fig. 2 *Neuron $N_{\theta,\nu}$ with threshold θ, output function $\mu(x)$ and weight vector* **w.**

The structure of recurrent neural network constructed for a set of rules Assume a non-void set of rules $\boldsymbol{R} = \{R_1, \ldots, R_q\}$ consisting of predicates $P_1, \ldots, P_n$. Assume there are m predicates that occur only in the rule antecedents. These predicates are called a-predicates. The rest of $n - m$ predicates occurring in rule consequents are called c-predicates. For rule set $\boldsymbol{R}$ we construct a layered recurrent neural network $\boldsymbol{N}_R$ with two hidden layers as follows.

1. The input layer consists of n receptors. One different predicate from $P_1, \ldots, P_n$ is assigned to each receptor.

2. The first hidden layer consists of q neurons of the type $N_{\theta,0}$ called C-neurons. A different rule $R \in \boldsymbol{R}$ is assigned to each C-neuron. Assume that a rule $R = L_1 \wedge \cdots \wedge L_r \supset L$ is assigned to neuron C. Then C has n inputs, one input for each L_i, i=1,..., r. If $L_i = P_i$, then C is connected to receptor P_i with weight w_i =1. If $L_i = \neg P_i$ then C is connected to receptor P_i with weight w_i = -1. The threshold of C is set to $\theta = (p - 1/4)$, where p is the number of positive literals in the antecedent of the rule R.

3. As there are $(n - m)c$-predicates, the second hidden layer consists of $n - m$ neurons called S-neurons. All the neurons in this layer are of the type $N_{0,1/2}$. A different c-predicate is assigned to each S-neuron. Assume that the c-predicate P is assigned to the neuron S. Then the neuron S is connected with all those C-neurons that have in the consequents assigned to them rules R literals P or $\neg P$. The weight between neuron S and C-neuron is 1 if to the C-neuron a rule with consequent P is assigned and the weight is -1 if to the C-neuron a rule with consequent $\neg P$ is assigned.

4. The output layer consists of $n - m$ neurons called T-neurons, which have 2 inputs and are of the type $N_{1,1/4}$. A different c-predicate is assigned to each T-neuron. Assume that the c-predicate P is assigned to the T-neuron T. Then the one input of T is connected to S-neuron in the second hidden layer with the same assigned predicate P. The second input of T is connected to the receptor with the same assigned predicate P. Both T-neuron input weights are set to 1.

5. The network output is fed back into the network input. If a predicate P is assigned **to** the output neuron, then output of this neuron is fed back into receptor with the same assigned predicate P. The topology of the network is obvious from Fig. 3.

Network computation (relaxation) States of the network change at discrete time points t_0, t_1, t_2,... in the following way:

1. At time t_0 the initial values 0, 1/2 or 1 are put on the network input.

2. At time t_n the input values are transferred through network layers to the output layer. If, during the transfer, at least one from the following conditions $E1$ or $E2$ occurs, the computation stops indicating error.

   $E1$. There exists a S-neuron with input value 1 on its positive input as well as on its negative input. Here positive (negative) input is input with positive (negative) weight.

   $E2$. There exists a T-neuron with input values 0 and 1.

3. If no error has occurred, the network output values are fed back into the network input. If no receptor has changed its value, the network computation stops and the result of the network computation is on the network input layer. If at least one receptor has changed its value, then n is increased to $n+1$ and the computation continues.

4. Computing Consequences

Theorem 1 Assume a non-void set of rules $\boldsymbol{R}$ consisting of predicates $P_1, \ldots, P_n$ and a set of facts $\boldsymbol{F}$. Assume that the set $\boldsymbol{R} \cup \boldsymbol{F}$ is non-contradictory. Let $\boldsymbol{N}_R$ be a neural network constructed to $\boldsymbol{R}$ as it has been described above. Let us put the following initial values on the network input. For all i we put

x_i=1 if $P_i \in \boldsymbol{F}$,
x_i=0 if $\neg P_i \in \boldsymbol{F}$,
x_i=1/2 otherwise.

Then relaxation of the network will finish without error after less than p steps, where p is the number of inputs with initial value 1/2. After relaxation the following will hold:

1. x_i=1 if and only if $\boldsymbol{R}$, $\boldsymbol{F} \mapsto P_i$,

2. x_i=0 if and only if $\boldsymbol{R}$, $\boldsymbol{F} \mapsto \neg P_i$.

Thus the network $\boldsymbol{N}_R$ computes all literal IF-THEN consequences of $\boldsymbol{R} \cup \boldsymbol{F}$.

Proof Assume that on the receptors $P_1, \ldots, P_n$ are values 0, 1/2 or 1. The values 0 and 1 are interpreted as predicate truth-values false and true respectively.

For each predicate with truth-value 1 we create literal $L_i = P_i$ and for each literal with truth-value 0 we create literal $L_i = \neg P_i$. The set of all thus created

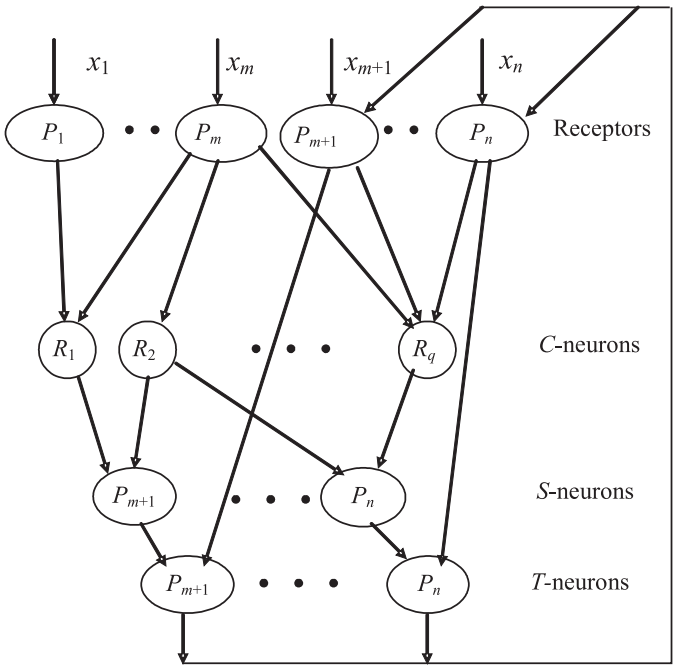

Fig. 3 *Recurrent neural network N_R constructed for a set of IF-THEN rules* $\boldsymbol{R}$.

literals we denote $\boldsymbol{G}$ and we call it input description. Suppose that the set of formulas $\boldsymbol{R}\cup\boldsymbol{G}$ is non-contradictory and that the network has carried out one relaxation step. We will prove the following assertions:

*A*1. Relaxation step will not stop on error.

*A*2. Only receptors with value 1/2 can change its input.

*A*3. Receptor P changes its value from 1/2 to 1 if and only if P can be deduced from $\boldsymbol{R}\cup\boldsymbol{G}$ using some rule $R \in \boldsymbol{R}$.

*A*4. Receptor P changes its value from 1/2 to 0 if and only if $\neg P$ can be deduced from $\boldsymbol{R}\cup\boldsymbol{G}$ using some rule $R \in \boldsymbol{R}$.

*A*5. Let $\boldsymbol{G'}$ denotes the input description after the relaxation step. Then the set of formulas $\boldsymbol{R}\cup\boldsymbol{G'}$ is non-contradictory.

Assume that a rule $R \in \boldsymbol{R}$, $R = L_1 \wedge \cdots \wedge L_r \supset L$ is assigned to a neuron C in the first hidden layer. If $L_i = P_i$, $i \in \{1, \ldots, r\}$, then the weight between receptor P_i and neuron C is 1. If $L_i = \neg P_i$, $i \in \{1,\ldots, r\}$, then the weight between receptor P_i and neuron C is -1. The scalar product $\mathbf{w}^T\mathbf{x}$ is maximal if all positive inputs into C (inputs with positive weights) are equal to 1 and all negative inputs into C

(inputs with negative weights) are equal to 0. In this case, $\mathbf{w}^T\mathbf{x} = p$, where p is the number of positive literals in the antecedent of the rule R. Hence the C-neuron postsynaptic potential is

$$(\mathbf{w}^T\mathbf{x} - \theta) = (p - (p - \frac{1}{4})) = \frac{1}{4}.$$

Recall that as the type of C-neuron is $N_{\theta,0}$ we are getting on its output $y = \mu(\mathbf{w}^T\mathbf{x} - \theta) = 1$. If at least one input value were 0 instead of being 1 or if it were 1 instead of being 0, then $(\mathbf{w}^T\mathbf{x} - \theta)$ would be less or equal to -3/4 and the output of C-neuron would be 0. Similarly, if at least one input value were 1/2, then $(\mathbf{w}^T\mathbf{x} - \theta)$ would be less or equal to $-1/4$ and the output of C-neuron would be 0. Therefore, the output of the neuron C is 1 only if the antecedent of the rule assigned to the C-neuron has truth-value 1.

Assume that S-neuron in the second hidden layer with assigned predicate P is connected to C-neurons $C_1, \ldots, C_s$ of the first hidden layer. Then consequents of rules assigned to $C_1, \ldots, C_s$ are P or $\neg P$. The input coming from C_i, i= 1, $\ldots$, s has weight 1 if the consequent of the rule R assigned to C_i is P and it has weight -1 if the consequent of the rule R assigned to C_i is $\neg P$. If S-neuron had input value 1 on its positive input as well as on its negative input, then it would be possible to deduce both P and $\neg P$ from $\boldsymbol{R}\cup\boldsymbol{G}$. However, it is not possible as the set $\boldsymbol{R}\cup\boldsymbol{G}$ is non-contradictory. Therefore, error $E1$ cannot occur and only the following three alternatives can come into being. (Recall that S-neurons can have only values 0 or 1 on their inputs and that S-neurons are of the type $N_{0,1/2}$.)

a) At least one positive input into S has value 1. In this case, all negative inputs must have value 0. Hence, $(\mathbf{w}^T\mathbf{x} - \theta) \geq 1$ and the output of S is 1.

b) At least one negative input into S has value 1. In this case, all positive inputs must have value 0. Hence, $(\mathbf{w}^T\mathbf{x} - \theta) \leq$ -1 and the output of S is 0.

c) All inputs into S are 0. In this case, $(\mathbf{w}^T\mathbf{x} - \theta) = 0$ and the output of S is 1/2.

Assume that c-predicate P is assigned to T-neuron of the output layer. Then the T-neuron, which is of the type $N_{1,1/4}$, is connected with receptor and S-neuron of the second hidden layer with the same assigned c-predicate P. The state just before the output of the network is fed back into network input can be described with one of the following five alternatives.

1. The output of receptor P is 1. If output of S-neuron were 0, then at least one negative input into this S-neuron would have value 1. Hence $\neg P$ could be deduced from $\boldsymbol{R}\cup\boldsymbol{G}$, which is not possible. Hence, the output of S-neuron must be at least 1/2 and for postsynaptic potential of T-neuron we get

   $(\mathbf{w}^T\mathbf{x} - \theta) \geq$(1+1/2-1)=1/2

   and the output of T neuron is 1. Thus, the receptor P will not change its value in the next computation step.

2. The output of receptor P is 0. If output of S-neuron were 1, then at least one positive input into this S-neuron would have value 1. Hence, P could be deduced from $\boldsymbol{R}\cup\boldsymbol{G}$, which is not possible. Hence, the output of S-neuron must be at most $1/2$ and for postsynaptic potential of T-neuron we get

   $(\mathbf{w}^T\mathbf{x}-\theta) \leq$(1/2-1)= -1/2

   and the output of T neuron is 0. Thus, the receptor P will not change its value in the next computation step.

3. The output of receptor P is $1/2$ and the output of S-neuron is 1 (i.e. the validity of P can be deduced using rule assigned to the S-neuron). Then the postsynaptic potential of T-neuron is

   $(\mathbf{w}^T\mathbf{x}-\theta)$=(1/2+1-1)= +1/2

   and the output of T neuron is 1. In the next computation step the value of P will be 1.

4. The output of receptor P is $1/2$ and the output of S-neuron is 0 (i.e. the validity of $\neg P$ can be deduced using rule assigned to the S-neuron). Then the postsynaptic potential of T-neuron is

   $(\mathbf{w}^T\mathbf{x}-\theta)$=(1/2-1)= -1/2

   and the output of T neuron is 0. In the next computation step the value of P will be 0.

5. The output of receptor P is $1/2$ and the output of S-neuron is $1/2$. Then the postsynaptic potential of T-neuron is

   $(\mathbf{w}^T\mathbf{x}-\theta)$=(1/2+1/2-1)= 0

   and the output of T neuron is $1/2$. In the next computation step the value of P will not change.

We have already shown that error $E1$ cannot occur. From 1 and 2 it follows that neither error $E2$ can occur. Therefore, assertion $A1$ is true.

Assertions $A2$-$A5$ follow immediately from 1-5.

From the above analysis we see that if the computation of the network starts from the input description $\boldsymbol{F}$ and if the set $\boldsymbol{R}\cup\boldsymbol{F}$ is non-contradictory, then it cannot stop on error (see $A1$, $A5$) and it must finish after less than p steps (see $A2$).

Let us define description $\boldsymbol{C}$ of network computation from $\boldsymbol{R}\cup\boldsymbol{F}$. $\boldsymbol{C}$ is a sequence of literals defined recursively:

a) If there are m facts in $\boldsymbol{F}$, the first m members of $\boldsymbol{C}$ are those facts.

b) Assume that in the n-th computation step r receptors P_i change their values from $1/2$ to 1 and s receptors P_i' change their values from $1/2$ to 0. Then we add to the current sequence $\boldsymbol{C}$ $r+s$ new members P_i and $\neg P_i'$.

Obviously, the sequence $\boldsymbol{C}$ is finite and due to $A3$ and $A4$ if $L \in \boldsymbol{C}$, then the subsequence $C_1,\ldots,C_p = L$ is IF-THEN deduction of L from $\boldsymbol{R}\cup\boldsymbol{F}$.

Assume that after computation the input x_i of receptor P_i equals to 1. Then $P_i \in \boldsymbol{C}$ and, therefore, $\boldsymbol{R}$, $\boldsymbol{F} \mapsto P_i$ holds. Similarly, if x_i=0, then $\neg P_i \in \boldsymbol{C}$ and therefore $\boldsymbol{R}$, $\boldsymbol{F} \mapsto \neg P_i$ holds.

On the other hand let $D_1, \ldots, D_p = L$ be an IF-THEN deduction of L from $\boldsymbol{R} \cup \boldsymbol{F}$. Then all D_i including $D_p = L$ must be members of description $\boldsymbol{C}$. According to b) if $L = P_i$ then x_i=1 must hold and if $L = \neg P_i$ then x_i =0 must hold. This completes the proof.

Example 1 Assume a set of rules $\boldsymbol{R}$ =$\{R_1, R_2, R_3, R_4\}$, where

R_1= $(\neg P_1 \wedge P_2) \supset P_4$,
R_2= $(\neg P_2 \wedge P_3) \supset \neg P_4$,
R_3= $(\neg P_1 \wedge P_4) \supset \neg P_3$,
R_4= $(\neg P_1 \wedge \neg P_3) \supset P_5$.

Let us construct network $\boldsymbol{N}_R$ for the set of rules $\boldsymbol{R}$. The rules of $\boldsymbol{R}$ consist of 5 predicates P_1, P_2, P_3, P_4, P_5, from which P_1, P_2 are a-predicates and P_3, P_4, P_5 are c-predicates. Therefore, the input layer will have 5 receptors with assigned predicates P_1, P_2, P_3, P_4, P_5. The first hidden layer will have 4 C-neurons with assigned rules R_1, R_2, R_3, R_4. The second hidden layer will have 3 S-neurons with assigned c-predicates P_3, P_4, P_5. The output layer will have 3 T-neurons with the same assigned c-predicates P_3, P_4, P_5. The configuration of the network is given in Fig. 4.

Network computation for the set of facts $\boldsymbol{F}$=$\{\neg P_1, P_2\}$ is given in Tab. I. From this table we can see that the set of all IF-THEN consequences of $\boldsymbol{R} \cup \boldsymbol{F}$ is $\{\neg P_1, P_2, \neg P_3, P_4, P_5\}$. For all of them there exists an IF-THEN deduction from $\boldsymbol{R} \cup \boldsymbol{F}$. For example the sequence of formulas $\boldsymbol{D}$ = $(\neg P_1, P_2, P_4, \neg P_3)$ is IF-THEN deduction of $\neg P_3$ from $\boldsymbol{R} \cup \boldsymbol{F}$.

The sequence of computation steps of the network if another set of facts $\boldsymbol{F}$=$\{\neg P_1, P_4\}$ is used can be found in Tab. II. The set of all IF-THEN consequences from $\boldsymbol{R} \cup \boldsymbol{F}$ is $\{\neg P_1, \neg P_3, P_4, P_5\}$. Since interpretations $\{\neg P_1, P_2, \neg P_3, P_4, P_5\}$ and $\{\neg P_1, \neg P_2, \neg P_3, P_4, P_5\}$ are both models of $\boldsymbol{R} \cup \boldsymbol{F}$, neither $\boldsymbol{R}$, $\boldsymbol{F} \vdash P_2$ nor $\boldsymbol{R}$, $\boldsymbol{F} \vdash \neg P_2$ hold.

Lemma 2 Assume that a set of literals $\{L_1, \ldots, L_r\}$ is added to $\boldsymbol{F}$. If the network computation stops on error, then $\boldsymbol{R}$, $\boldsymbol{F} \vdash \neg(\ L_1 \wedge \cdots \wedge L_r)$ holds.

Proof If the network computation stops on error, then the set $\boldsymbol{R} \cup \boldsymbol{F} \cup \{L_1, \ldots, L_r\}$ must be contradictory according to Theorem 1. As the set $\boldsymbol{R} \cup \boldsymbol{F}$ is not contradictory, it has at least one model. However, in no model of $\boldsymbol{R} \cup \boldsymbol{F}$ the formula $(\ L_1 \wedge \cdots \wedge L_r)$ can be valid. Therefore, in all models of $\boldsymbol{R} \cup \boldsymbol{F}$ formula $\neg(\ L_1 \wedge \cdots \wedge L_r)$ must be valid and, consequently, $\boldsymbol{R}$, $\boldsymbol{F} \vdash \neg\ (\ L_1 \wedge \cdots \wedge L_n\ )$ holds.

Example 2 Assume a set of rules $\boldsymbol{R}$ =$\{R_1, R_2, R_3\}$, where

R_1= $(P_1 \wedge \neg P_4) \supset P_2$,
R_2= $(\neg P_2 \wedge \neg P_3) \supset P_1$,
R_3= $(\neg P_4 \wedge P_5) \supset \neg P_3$.

Let us construct network $\boldsymbol{N}_R$ for the set of rules $\boldsymbol{R}$. The rules consist of 5 predicates P_1, P_2, P_3, P_4, P_5, from which P_4, P_5 are a-predicates and P_1, P_2, P_3

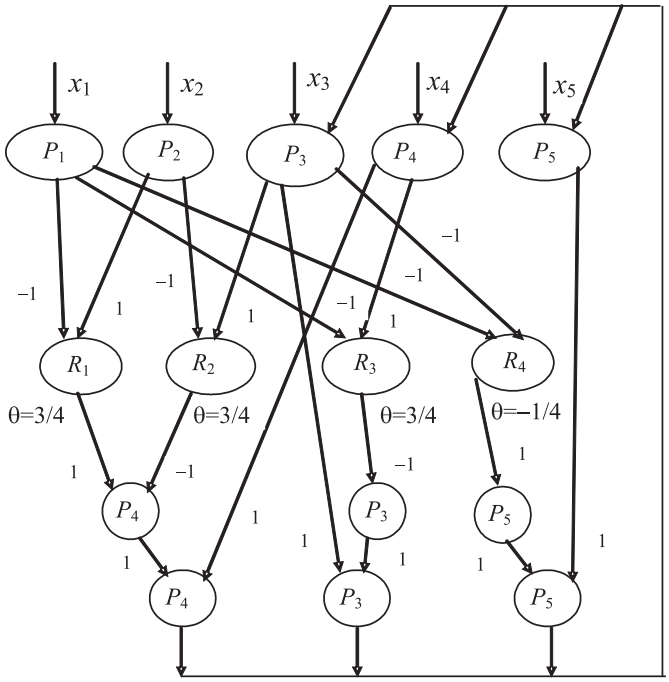

Fig. 4 *Recurrent network constructed for a rule set of Example 1.*

n	x_1	x_2	x_3	x_4	x_5
0	0	1	1/2	1/2	1/2
1	0	1	1/2	1	1/2
2	0	1	0	1	1/2
3	0	1	0	1	1
4	0	1	0	1	1

Tab. I *Example 1: Network computation from the set of facts* $\boldsymbol{F}$=$\{\neg P_1, P_2\}$.

n	x_1	x_2	x_3	x_4	x_5
0	0	1/2	1/2	1	1/2
1	0	1/2	0	1	1/2
2	0	1/2	0	1	1
3	0	1/2	0	1	1

Tab. II *Example 1: Network computation from the set of facts* $\boldsymbol{F}$=$\{\neg P_1, P_4\}$.

are c-predicates. Therefore, the input layer will have 5 receptors with assigned predicates P_1, P_2, P_3, P_4, P_5. The first hidden layer will have 3 C-neurons with assigned rules R_1,R_2,R_3. The second hidden layer will have 3 S-neurons with assigned c-predicates P_1, P_2, P_3. The output layer will have 3 T-neurons with the same assigned c-predicates P_1, P_2, P_3. The configuration of the network is given in Fig. 5.

Network computation for the set of facts $\boldsymbol{F}=\{\neg P_4,\ P_5\}$ is given in Tab. III. From this table we can see that the set of all IF-THEN consequences of $\boldsymbol{R}\cup\boldsymbol{F}$ is $\{\neg P_3,\ \neg P_4,\ P_5\}$.

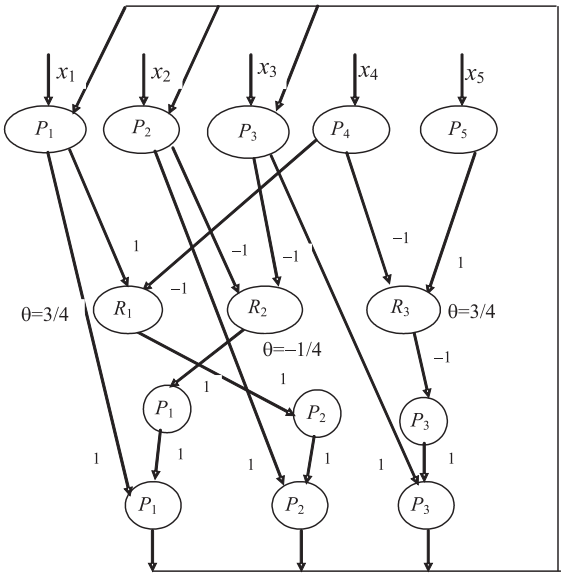

Fig. 5 *Recurrent network constructed for a rule set of Example 2.*

According to Lemma 1 literals $\neg P_3$, $\neg P_4$, P_5 are also logical consequences of $\boldsymbol{R}\cup\boldsymbol{F}$. Using Lemma 2, we may try to find some further literal logical consequences of $\boldsymbol{R}\cup\boldsymbol{F}$. Let us enlarge set of facts by adding the literal $\neg P_2$ and repeat computation of the network. The result is given in Tab. IV. The computation has stopped at the third step with error $E2$. Hence, according to Lemma 2 literal P_2 is logical consequence of $\boldsymbol{R}\cup\boldsymbol{F}$. It is easy to see that both interpretations (P_1, P_2, $\neg P_3$, $\neg P_4$, P_5), ($\neg P_1$, P_2, $\neg P_3$, $\neg P_4$, P_5) are models of $\boldsymbol{R}\cup\boldsymbol{F}$. Hence, neither P_1 nor $\neg P_1$ may be logical consequences of $\boldsymbol{R}\cup\boldsymbol{F}$.

Lemma 3 Assume that an interpretation of predicates $\boldsymbol{I}$ is on the input of the network $\boldsymbol{N}_R$.

1. If $\boldsymbol{I}$ is a model of $\boldsymbol{R}$, then network computation stops after the first iteration step without generating error.
2. If $\boldsymbol{I}$ is not a model of $\boldsymbol{R}$, then the network computation stops during the first iteration step with error.

n	x_1	x_2	x_3	x_4	x_5
0	1/2	1/2	1/2	0	1
1	1/2	1/2	0	0	1
2	1/2	1/2	0	0	1

Tab. III *Example 2: Network computation from the set of facts* $\boldsymbol{F}$=$\{\neg P_4, P_5\}$.

n	x_1	x_2	x_3	x_4	x_5
0	1/2	0	1/2	0	1
1	1/2	0	0	0	1
2	1	0	0	0	1
3		$E2$			

Tab. IV *Example 2: Network computation for the set of facts* $\boldsymbol{F}$=$\{\neg P_2, \neg P_4, P_5\}$.

Proof

1. If interpretation $\boldsymbol{I}$ is a model of $\boldsymbol{R}$, then according to the Theorem 1 the network computation must finish without error. During the first iteration step no receptor will change its value, because only receptors with value 1/2 can change their values (see property $A2$ in the proof of Theorem 1). Hence, the computation stops after the first step.

2. If $\boldsymbol{I}$ is not a model of $\boldsymbol{R}$, then at least one rule $R \in \boldsymbol{R}$ must be invalid in $\boldsymbol{I}$. The consequent of this R may be either P or $\neg P$. Assume that the consequent is P. Then initial output value of the receptor with assigned predicate P must be 0. As antecedent of R must be valid in $\boldsymbol{I}$, the output of C-neuron with assigned rule R is 1. Assume that the C-neuron is connected with S-neuron S. The weight of this connection is 1. If some negative input of S had also value 1, then error $E1$ would occur. If no such negative input exists, then the output of S must be 1. The output of S is connected to the T-neuron T with assigned predicate P. The second input of T must come from receptor P. Therefore, on the input of T must be values 0 and 1. Consequently, the error $E2$ must occur. For the case that the consequent of R is $\neg P$ the proof can be completed similarly. Thus, if interpretation $\boldsymbol{I}$ is not model of $\boldsymbol{R}$, then during the first iteration step network computation stops with error.

5. Conclusion

Let $\boldsymbol{D}_0 = \{L\colon\ \boldsymbol{R},\ \boldsymbol{F} \mapsto L\}$ be the set of all literal IF-THEN consequences from $\boldsymbol{R} \cup \boldsymbol{F}$ and let $\boldsymbol{D} = \{L\colon \boldsymbol{R},\ \boldsymbol{F} \vdash L\}$ be a set of all literal logical consequences from $\boldsymbol{R} \cup \boldsymbol{F}$. We have shown (see Lemma 1 and Theorem 1) that

1. $\boldsymbol{F} \subset \boldsymbol{D}_0 \subset \boldsymbol{D}$.

2. Network $\boldsymbol{N}_R$ can be used for determination of all literals from $\boldsymbol{D}_0$.

Thus, the set $\boldsymbol{D}_0$ may be considered to be an approximation of $\boldsymbol{D}$. Moreover, the approximation $\boldsymbol{D}_0$ may be easily refined using Lemma 2. We may proceed as follows. We enlarge the set of facts by adding a particular literal L_i to the set of facts. If during the following computation an error arises, then $\neg L_i$ is logical consequence of $\boldsymbol{R} \cup \boldsymbol{F}$.

For example, assume that after computation of $\boldsymbol{D}_0$, the value of predicate P_1 was 1/2. Assume that we enlarged the set of facts $\boldsymbol{F'} = \boldsymbol{F} \cup \{\neg P_1\}$ and that the computation finished with error. Then according to Lemma 2, P_1 must be in $\boldsymbol{D}$. Therefore, the initial approximation $\boldsymbol{D}_0$ of $\boldsymbol{D}$ may be refined by adding literal P_1.

The open question is if $\boldsymbol{N}_R$ can be used for determination of all literals from $\boldsymbol{D}$. The answer is positive. However, the procedure might be very time-consuming. The procedure may proceed as follows. At first, the network carries out the computation to determine all literals belonging to $\boldsymbol{D}_0$. Then for those predicates that are assigned to receptors with value 1/2, all the possible combinations of 0 and 1 are subsequently generated. Generated combinations are subsequently put on the network input and the network carries out the computation. As only values 0 and 1 are on receptors, the network input constitutes an interpretation of predicates $\boldsymbol{I}$. According to Lemma 3 the network carries out only one iteration step. If, during the iteration step, an error is generated, the interpretation $\boldsymbol{I}$ is not model of $\boldsymbol{R}$ and it need not to be considered. If no error is generated, then $\boldsymbol{I}$ is a model of $\boldsymbol{R}$. After all combinations are generated and computed, we can assert the following:

1. $P \in \boldsymbol{D}$ holds, if input x of the receptor P has been equal to 1 in all models of $\boldsymbol{R}$.

2. $\neg P \in \boldsymbol{D}$ holds, if input x of the receptor P has been equal to 0 in all models of $\boldsymbol{R}$.

3. Neither $P \in \boldsymbol{D}$ nor $\neg P \in \boldsymbol{D}$ hold, if input x of the receptor P has been equal to 1 in some models of $\boldsymbol{R}$ and it has been equal to 0 in some others.

We have explored the IF-THEN deduction from facts, which is commonly used in everyday life or in expert reasoning. We have proved that the IF-THEN deduction could be easily realized with a simple layered recurrent neural network with two hidden layers and three level threshold neurons. If, at the beginning, truth-values of p-predicates are unknown, then the network will determine the set of all literal IF-THEN consequences $\boldsymbol{D}_0$ after p iteration steps at maximum (see Theorem 1). By means of Lemma 2 we may simulate proof of contradiction and we can get further literal logical consequences. If, after determination of $\boldsymbol{D}_0$, the truth-values of q-predicates remain unknown, we need at maximum $2q(q\text{-}1)$ iteration steps for systematic application of proof of contradiction (recall that for each predicate with unknown truth-value we must postulate at first P and then $\neg P$). It is also possible to get all the logical consequences. If, after application of proof of contradiction, r-predicates remain unknown, then we need 2^r further iteration steps. Hence, for determination of all literal logical consequences $p+2q(q-1)+2^r$ iteration steps are needed at maximum, which can be quite time-consuming for large r.

References

[1] Anderson J. R., Lebiere C.: Atomic components of thought, Hillsdale, 1998.

[2] Kurfess F.: Neural Networks and Structured Knowledge: Knowledge representation and reasoning, Applied Intelligence, **11**, 1999, pp. 5-13.

[3] Towell G. G., Shavlik J. W.: Knowledge-based artificial neural networks, Artificial Intelligence, **70**, 1994, pp. 119-165.

[4] Hitzler P., Holldobler S., Seda A. K.: Logic programs and connections networks, Journal of Applied Logic, **2**, 2004, pp. 245-272.

[5] Holldobler S., Kalinke Y., Storr H. P.: Approximating the semantics of logic programs by recurrent neural networks, Applied Intelligence, **11**, 1999, pp. 45-58.

[6] Stenning K., Van Lambalgen M.: Human reasoning and cognitive science, MIT Press, 2008.

[7] Mendelson E.: Introduction to mathematical logic, CRC Press, 2001.