On the Use of Fuzzy Logic and Learning Automata Optimization to Resolve the Liar and Related Paradoxes

Ath. KEHAGIAS a,1, and G. KARTSIOTIS b

a School of Engineering, Aristotle University, Thessaloniki, Greece; E-mail: kehagiat@auth.gr.
b School of Engineering, Aristotle University, Thessaloniki, Greece.

Abstract. We show that logical paradoxes based on self-reference (of which the Liar is the best known example) are equivalent to the non-existence of solutions to a numerical system of equations, the so-called truth-value equations. Furthermore, we show that in many cases a self-referential system which does not possess a crisp (Boolean) solution can be solved by expanding the solution set to include fuzzy solutions. Then we formulate the computation of these fuzzy solutions as an optimization problem and, by numerical experiments, we demonstrate that teams of Learning Automata (of a type intermediate between finite action and continuous action automata) can be successfully used to solve the optimization problem. In this manner, the combination of fuzzy logic and learning automata resolves a wide class of paradoxes.

Keywords. Fuzzy logic, learning automata, paradox, liar, optimization.

1. Introduction

In this paper we study self-referential systems, i.e., collections of sentences which talk about each other. The classical example is the "Liar Sentence":

"This sentence is false". (1)

Self-reference becomes more obvious if we rewrite (1) as follows:

\[ A = "A is false". \] (2)

Many similar self-referential systems can be constructed and it is well known that some of them generate logical paradoxes. For example, in (2) if \( A \) is true, then what it says must hold, i.e. \( A \) must be false. Then the opposite of \( A \), i.e. "\( A \) is true" must be true, but then what \( A \) says is true, i.e. "\( A \) is false". This reasoning produces an oscillation between two conclusions: first that \( A \) is false, then that it is true. A similar oscillation would be obtained if the starting assumption were that \( A \) is false. The well-known Liar Paradox is that, by the above analysis, \( A \) can be neither true nor false; in other words it does not have a (classical) truth value. Another example of two sentences which talk about each other is the "Inconsistent Dualist":

\[ A_1 = "A_2 is true", \quad A_2 = "A_1 is false". \] (3)

Similarly to the Liar sentence, a paradox of non-existence of truth values arises in connection to the system (3).

In this paper we use Learning Automata (LA) optimization to compute consistent fuzzy truth values for a wide class of self-referential systems, including "paradoxical" systems of the type mentioned in the previous paragraph. We show that the use of fuzzy logic and learning automata offers a resolution of a wide class of paradoxes.

The current paper is based on previous work by Zadeh [34], Grim et al. [11,18] and ourselves [15,33]. Namely, in [15,33] we have constructed a large family of self-referential systems and we have shown that every such self-referential system generates a system of logical equations which, in turn, generate a system of (numerical) truth value equations. Some of the systems in the aforementioned family generate paradoxes, which always reduce to the nonexistence of a classical (crisp) solution to the corresponding truth value equations. But, as we have shown in [15,33], if the set of admissible solutions is expanded to include fuzzy truth values, then a solution is always available and the paradox disappears. The paper is organized as follows. In

1Corresponding Author: Ath. Kehagias, School of Engineering, Aristotle University, Thessaloniki, Greece; E-mail: kehagiat@auth.gr.
2Because of space limitations we do not present an extensive literature review in the current paper. The interested reader will find many references to the extensive bibliography on the Liar paradox and self-referential systems in [15,33].
Section 2 we introduce numerical truth value equations for the resolution of logical paradoxes. In Section 3 we show how solutions to the truth value equations can be computed using LA optimization. In Section 4 we validate our approach by numerical experimentation. Concluding remarks and future research directions appear in Section 5. In Appendix A we speculate on some connections between the current work, Game Theory and Psychology.

2. Self-referential Systems, Paradoxes and Systems of Equations

In this section we review our previous results on self-referential systems and paradoxes. We show that the previously discussed self-referential systems (the Liar and the Dualist) can be rewritten as sentences which talk about their own truth values. These sentences can be understood as a system of logical equations and an associated system of numerical equations (the truth value equations) can be readily obtained. We next show that this approach can be generalized to a wide class of self-referential systems and argue that a self-reference paradox can be resolved by solving the corresponding numerical equation; this is not always possible in the context of classical, crisp logic but, in the context of fuzzy logic, every self-referential system (subject to some mild conditions) possesses at least one solution.

We start with the two examples previously discussed. Both the Liar and the Inconsistent Dualist can be expressed in terms of statements about truth values. In anticipation of later developments, we denote “True” by 1 and “False” by 0 (at this point 1 and 0 should be understood as mere symbols, rather than numbers). Now consider the following examples.

Example 1: The Liar. The Liar sentence (2) can be restated as follows: \( A = \text{“The truth value of } A \text{ is 0”} \) or, more compactly,

\[ A = \text{“Tr}(A) = 0” \]  \hspace{1cm} (4)

where “\( \text{Tr}(A) = a \)” means “The truth value of \( A \) is \( a \)”, \( A \) is the Liar sentence and \( a \in \{0, 1\} \).

Example 2: The Inconsistent Dualist. Similarly, the Inconsistent Dualist (3) can be written as follows

\[ A_1 = \text{“Tr}(A_2) = 1”, \quad A_2 = \text{“Tr}(A_1) = 0”. \]  \hspace{1cm} (5)

We can generalize the above examples, i.e. we can form sentences which are similar to (4) and (5) but quite more complex. We will show how this can be done by example; the interested reader can consult [15,33] for an extended and more rigorous exposition of our approach. In what follows \( \wedge \) stands for “and”, \( \vee \) stands for “or” and \( \neg \) stands for “not”.

Example 3: The Consistent Dualist. Consider the pair of sentences: \( A_1 = \text{“} A_2 \text{ is true”}, \ A_2 = \text{“} A_1 \text{ is true”} \). This can be written as

\[ A_1 = \text{“Tr}(A_2) = 1”, \quad A_2 = \text{“Tr}(A_1) = 1”. \]  \hspace{1cm} (6)

Example 4. The self-referential system

\[ A_1 = \text{“} A_2 \text{ is true and } A_3 \text{ is false”} \]  \hspace{1cm} (7)
\[ A_2 = \text{“} A_1 \text{ is true and } A_3 \text{ is false”} \]  \hspace{1cm} (8)
\[ A_3 = \text{“} A_1 \text{ is false”}. \]  \hspace{1cm} (9)

is written as

\[ A_1 = \text{“Tr}(A_2) = 1” \wedge \text{“Tr}(A_3) = 0” \]  \hspace{1cm} (10)
\[ A_2 = \text{“Tr}(A_1) = 1” \wedge \text{“Tr}(A_3) = 0” \]  \hspace{1cm} (11)
\[ A_3 = \text{“Tr}(A_1) = 0”. \]  \hspace{1cm} (12)

Example 5. Consider

\[ A_1 : \text{“} A_2 \text{ has truth value 0.90”} \]
and “\( A_3 \) has truth value 0.20” \hspace{1cm} (13)
\[ A_2 : \text{“} A_1 \text{ has truth value 0.80”} \]
and “\( A_3 \) has truth value 0.30” \hspace{1cm} (14)
\[ A_3 : \text{“} A_1 \text{ has truth value 0.10”}. \]  \hspace{1cm} (15)

(13)-(15) can be written as

\[ A_1 = \text{“Tr}(A_2) = 0.90” \wedge \text{“Tr}(A_3) = 0.20” \]  \hspace{1cm} (16)
\[ A_2 = \text{“Tr}(A_1) = 0.80” \wedge \text{“Tr}(A_3) = 0.30” \]  \hspace{1cm} (17)
\[ A_3 = \text{“Tr}(A_1) = 0.10”. \]  \hspace{1cm} (18)

Example 6. We conclude with a more complex example:

\[ A_1 : \text{“(} A_1 \text{ has truth value 0.75” and “} A_2 \text{ has truth value 0.35”) or “} A_4 \text{ has truth value 1.00”} \]
\[ A_2 : \text{“} A_1 \text{ or } A_3 \text{ has truth value 1.00”} \]
and “\( A_4 \) has truth value 0.10” \hspace{1cm} (19)
\[ A_3 : \text{“} A_2 \text{ has truth value 0.00” and “} A_3 \text{ has truth value 0.35”} \]
\[ A_4 : \text{“The opposite of } A_1 \text{ has truth value 0.25”,} \]

which translates to
\[ A_1 = (C_1 \land C_2) \lor C_3 \] (19)
\[ A_2 = C_4 \land C_5 \] (20)
\[ A_3 = C_6 \land C_7 \] (21)
\[ A_4 = C_8 \] (22)

with

\[ C_1 : \text{"The truth value of } A_1 \text{ is 0.75"} \]
\[ C_2 : \text{"The truth value of } A_2 \text{ is 0.35"} \]
\[ C_3 : \text{"The truth value of } A_4 \text{ is 1.00"} \]
\[ C_4 : \text{"The truth value of } A_1 \lor A_3 \text{ is 1.00"} \]
\[ C_5 : \text{"The truth value of } A_2 \text{ is 0.10"} \]
\[ C_6 : \text{"The truth value of } A_3 \text{ is 0.00"} \]
\[ C_7 : \text{"The truth value of } A_4 \text{ is 0.35"} \]
\[ C_8 : \text{"The truth value of } A'_1 \text{ is 0.25"} . \]

In all of the above examples we have dealt with systems of logical equations, in which the following symbols appear: \( A_1, A_2, \ldots \) to denote the self-referential sentences, “\( \lor \)”, “\( \land \)”, “\( \lnot \)” to denote the usual logical operators or, and, not, and the expression “\( \text{Tr}(B) = b \)” to denote “The truth value of \( B \) is \( b \)”. Using these symbols as building blocks we can build a wide family of systems of self-referential sentences of the form

\[ A_1 = F_1(A_1, \ldots, A_M) \]
\[ A_2 = F_2(A_1, \ldots, A_M) \]
\[ \ldots \]
\[ A_M = F_M(A_1, \ldots, A_M). \] (23)

where \( F_1(\cdot), F_2(\cdot), \ldots, F_M(\cdot) \) are logical formulas. For each logical system of the form (23), a system of numerical equations can be obtained by providing numerical interpretations for the logical connectives \( \land, \lor, \lnot \), and the function \( \text{Tr}(B) = b \).

Numerical implementations of \( \land, \lor, \lnot \) are well known in the context of fuzzy logic. If 0, 1 are understood as numbers (rather than mere symbols) then \( \land, \lor \) can be understood as numerical functions which satisfy certain conditions: compatibility with the classical (crisp) truth tables (of conjunction and disjunction respectively), symmetry, associativity and monotonicity. Such functions (\( t\)-norms and \( t\)-conorms) have been extensively studied by fuzzy logicians [16]. Similar things hold for the negation. Several typical implementations are presented in Table 1. Each row in the table corresponds to a family of logical connectives (the families listed in Table 1 are called: standard, algebraic, bounded and drastic). When restricted to the classical truth values 0 and 1, these implementations are identical to the classical logical operations of conjunction, disjunction and negation.

<table>
<thead>
<tr>
<th>( x \land y )</th>
<th>( x \lor y )</th>
<th>( x' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>min(( x, y ))</td>
<td>max(( x, y ))</td>
<td>1-( x )</td>
</tr>
<tr>
<td>( xy )</td>
<td>( x + y - xy )</td>
<td>1-( x )</td>
</tr>
<tr>
<td>( y ) when ( x = 1 )</td>
<td>( x ) when ( y = 0 )</td>
<td>1-( x )</td>
</tr>
<tr>
<td>( 0 ) else</td>
<td>( y ) when ( x = 0 )</td>
<td>( 1 ) else</td>
</tr>
</tbody>
</table>

Table 1

We now turn to “\( \text{Tr}(B) = b' \)”. It is a logical statement which talks about the sentence \( B \) (more precisely: about \( \text{Tr}(B) \)), which is the truth value of \( B \) and the number \( b' \); hence it can be assigned a fuzzy truth value \( v \) in the interval \([0,1]\). When \( \text{Tr}(B) \) is equal to \( b \), then \( v \) must achieve the maximum value, \( v = 1 \) (since “\( \text{Tr}(B) = b' \)” is true); in general \( v \) must be a decreasing function of the absolute difference \( |\text{Tr}(B) - b| \). These requirements can be satisfied by setting

\[ v = \text{Tr}("\text{Tr}(B) = b'") = 1 - |\text{Tr}(B) - b| \] (24)

which we will use in the sequel; (24) has been used by other authors as well [11].

Now consider a sentence \( F(A_1, \ldots, A_M) \) which talks about the truth values of \( A_1, \ldots, A_M \). Replacing \( A_m \) with \( \text{Tr}(A_m) \) (for all \( m \in \{1, 2, \ldots, M\} \)) and understanding the symbols \( \land, \lor, \lnot \) as numerical functions (according to Table 1) we obtain a numerical function

\[ f(\text{Tr}(A_1), \ldots, \text{Tr}(A_M)) \]

with domain \([0,1]^M\) and range \([0,1]\).

Given a self-referential system of the form (23), for each \( m \in \{1, 2, \ldots, M\} \) we perform the above procedure on \( F_m(A_1, \ldots, A_M) \); also, for simplicity of notation we replace \( \text{Tr}(A_m) \) by \( x_m \) (for \( m \in \{1, 2, \ldots, M\} \)). Then (23) yields a system of \( M \) numerical equations in \( M \) unknowns:

\[ x_1 = f_1(x_1, \ldots, x_M) \]
\[ x_2 = f_2(x_1, \ldots, x_M) \]
\[ \ldots \]
\[ x_M = f_M(x_1, \ldots, x_M). \] (25)

We will refer to (25) as the system of truth value equations.
To illustrate we continue Examples 1–6.

**Example 1 Continued:** \( \text{Tr}(A) = 1 - |\text{Tr}(A) - 0| \Rightarrow x = 1 - |x - 0| \Rightarrow \\
\quad x = 1 - x \)  
(26)

(26) is the truth value equation for the Liar. The Liar paradox consists of the fact that, in the context of classical logic the set of admissible truth values is \( \{0, 1\} \) and (26) has no solution in this set, i.e. the Liar sentence has no classical truth value. But in the fuzzy logic context the solution set is \([0, 1]\) and (26) has a unique solution \( x = 1/2 \). The Liar sentence has a fuzzy truth value (it is half-true) and the Liar paradox has been resolved.

**Example 2 Continued:** Similarly, for the Inconsistent Dualist, with \( x_m = \text{Tr}(A_m) \) \((m = 1, 2)\) we get
\[
\begin{align*}
x_1 &= 1 - |x_2 - 1|, \quad x_2 = 1 - |x_1 - 0| \\
\text{and, since } x_1, x_2 &\in [0, 1], \text{ we finally get the truth value equations}
\end{align*}
\]
\[
\begin{align*}
x_1 &= x_2 \quad (27) \\
x_2 &= 1 - x_1. \quad (28)
\end{align*}
\]

Obviously, eqs. (27)–(28) have the unique solution \( x = (x_1, x_2) = (1/2, 1/2) \).

**Example 3 Continued:** In this case the truth value equations are (cmp. (6))
\[
\begin{align*}
x_1 &= 1 - |x_2 - 1|, \quad x_2 = 1 - |x_1 - 1| \\
\text{which can be written in simple form as}
\end{align*}
\]
\[
\begin{align*}
x_1 &= x_2, \\
x_2 &= x_1. \quad (29)
\end{align*}
\]

Any vector of the form \( x = (u, u) \) \((u \in [0, 1])\) is a solution; i.e. there is an infinite number of consistent truth value assignments including complete truth \( \text{Tr}(A_1) = \text{Tr}(A_2) = 1 \) and complete falsity \( \text{Tr}(A_1) = \text{Tr}(A_2) = 0 \);

**Example 4 Continued:** If we implement \( \land \) by the min operator, the truth value equations become (cmp. (10)–(12))
\[
\begin{align*}
x_1 &= \min |x_2, (1 - x_3)| \quad (30) \\
x_2 &= \min |x_1, (1 - x_3)| \\
x_3 &= 1 - x_1
\end{align*}
\]

These can be solved analytically to obtain the general solution
\[
\bar{x} = (u, u, 1 - u)
\]

with \( u \in [0, 1] \). Note that this includes the extremal solutions \((1, 1, 0)\) and \((0, 0, 1)\) as well as the “mid-point solution” \((1/2, 1/2, 1/2)\). On the other hand, implementing \( \land \) by the product t-norm we obtain the truth value equations
\[
\begin{align*}
x_1 &= x_2 \cdot (1 - x_3) \\
x_2 &= x_1 \cdot (1 - x_3) \quad (31) \\
x_3 &= 1 - x_1
\end{align*}
\]

and, solving analytically we obtain the solutions
\[
(0, 0, 1), \quad (1, 1, 0), \quad (-1, 1, 0).
\]

(The last solution is inadmissible as a truth value assignment.) We see that the same logical system can lead to different truth value assignments, depending on the implementation of “AND” (and “OR”, as we will see presently).

**Example 5 Continued.** If we implement \( \land \) with the min operator the truth value equations become (cmp. (16)–(18))
\[
\begin{align*}
x_1 &= \min |1 - |x_2 - 0.90|, 1 - |x_3 - 0.20|) \\
x_2 &= \min |1 - |x_1 - 0.80|, 1 - |x_3 - 0.30| \quad (32) \\
x_3 &= 1 - |x_1 - 0.10|.
\end{align*}
\]

These equations cannot be further reduced and while in principle they can be solved analytically by distinguishing cases, this is quite tedious. The situation is similar when we implement \( \land \) by product, the truth value equations become
\[
\begin{align*}
x_1 &= (1 - |x_2 - 0.90|) \cdot (1 - |x_3 - 0.20|) \\
x_2 &= (1 - |x_1 - 0.80|) \cdot (1 - |x_3 - 0.30|) \quad (33) \\
x_3 &= 1 - |x_1 - 0.10|.
\end{align*}
\]

**Example 6 Continued.** The truth value equations under min/max implementation are (cmp. (19)–(22))
\[
\begin{align*}
x_1 &= \max |\min (1 - |x_1 - 0.75|, 1 - |x_2 - 0.35|), 1 - |x_4 - 1.00|) \\
x_2 &= \min |1 - \max (x_1, x_3) - 1|, 1 - |x_4 - 0.10| \quad (34) \\
x_3 &= \min |1 - |x_2 - 0.00|, 1 - |x_3 - 0.35| \\
x_4 &= 1 - |1 - x_1 - 0.25|.
\end{align*}
\]

Under product / sum implementation they are:
\[ x_1 = (1 - |x_1 - 0.75|) \cdot (1 - |x_2 - 0.35|) + (1 - |x_4 - 1.00|) \]
\[ - (1 - |x_1 - 0.75|) \cdot (1 - |x_2 - 0.35|) \cdot (1 - |x_4 - 1.00|) \]
\[ x_2 = (1 - |x_1 + x_3 - x_1 x_3 - 1.00|) \cdot (1 - |x_4 - 0.10|) \]
\[ x_3 = (1 - |x_2 - 0.00|) \cdot (1 - |x_3 - 0.35|) \]
\[ x_4 = 1 - |1 - x_1 - 0.25| . \]

The above examples illustrate several points. It is possible that a system of truth value equations admits no solution in \([0,1]^M\) but has one or more solutions in \([0,1]^M\). Hence a self-referential system which is paradoxical in the context of classical logic, may be resolved (i.e. become non-paradoxical) in the context of fuzzy logic. It is also possible that the truth value equations have more than one solutions (Examples 3 and 4); every such solution of (25) is a consistent truth value assignment and the self-referential system is indeterminate; this may be considered as a paradox or not. Finally, there are cases (Examples 5 and 6) where it is not clear whether one, many or no solution exists; however we have proved [15,33] that, under mild conditions, every self-referential system expressed in a language \(L\) (which is well defined in [15,33]) admits at least one consistent truth value assignment.

Hence every self-referential paradox of the type presented in the above examples can be removed in the context of fuzzy logic, i.e. it possesses at least one consistent fuzzy truth value assignment. To obtain this assignment the numerical system (25) must be solved; when this is not possible analytically, some computational method must be provided; in the next section we tackle this problem.

3. Solving the Truth Value Equations by Learning Automata Minimization

In this section we present a method for actually computing truth values (i.e. solving the truth value equations) using learning automata (LA) minimization [2, 21,25,26,31].

The general approach for equation solving by error minimization is well known; here we review it in connection to the truth value equations (25). We want values \(x_1, \ldots, x_M\) such that the left and right sides are equal in each of the equations (25). Let us define the partial inconsistency

\[ J_m(x_1, \ldots, x_M) = (x_m - f_m(x_1, \ldots, x_M))^2, \]  

(36)

for \(m \in \{1, 2, \ldots, M\}\); \(J_m\) takes values in \([0,1]\); it is the discrepancy between the postulated truth value \(x_m\) and the truth value resulting from evaluating \(f_m(x_1, \ldots, x_M)\). Let us also define the average inconsistency

\[ J(x_1, \ldots, x_M) = \frac{\sum_{m=1}^M (x_m - f_m(x_1, \ldots, x_M))^2}{M}; \]  

(37)

it also takes values in \([0,1]\). It is clear that

\[ J(x_1, \ldots, x_M) = \frac{\sum_{m=1}^M J_m(x_1, \ldots, x_M)}{M}. \]  

(38)

When \(J\) is large, (25) is not satisfied; when \(J\) is small, (25) is almost satisfied; a true solution \((x_1, \ldots, x_M)\) of (25) achieves the global minimum \(J(x_1, \ldots, x_M) = 0\). There may be more than one such \((x_1, \ldots, x_M)\); as remarked above, the existence of at least one is guaranteed. In short, a solution of (25) is a global minimizer \((x_1, \ldots, x_M)\) such that \(J(x_1, \ldots, x_M) = 0\). The minimizer can be achieved by various function minimization algorithms; in this paper we use several algorithms based on learning automata (LA).

A learning automaton [20] is a simple agent which probabilistically chooses an action (from a finite set of possible actions); the action probabilities are updated (learned) according to a response which is perceived as reward or punishment, depending on its effect on the goal of minimizing \(J\).

LA have often been used to solve optimization problems. A good exposition appears in the book [21]; see also the review paper [31]; some additional notable papers are [2,25,26]. For a system of \(M\) self-referential sentences we use a team of \(M\) automata, one automaton per sentence. We use variable structure automata of the type presented in [20, Chapters 4 and 8]; these evolve in discrete time \(t = 0,1,2,\ldots\) and are characterized by the following.

1. A vector of actions \([a_1(t), \ldots, a_M(t)]\): \(a_m(t)\) is the action taken by the \(m\)-th automaton at time \(t\), with \(m \in \{1,2,\ldots,M\}\) and \(a_m(t) \in \{1,2,\ldots,K\}\) (the set of actions is finite).
2. The corresponding, time evolving, action probabilities \(p_1^m(t), \ldots, p_K^m(t))\), where \(p_k^m(t) = \Pr(a_m(t) = k)\) and \(m \in \{1,2,\ldots,M\}\).
3. A vector of responses \([b_1(t), \ldots, b_M(t)]\): i.e. \(b_m(t)\) is the response to the action \(a_m(t)\) (\(m \in \{1,2,\ldots,M\}\)); \(b_m(t) \in \{0,1\}\), where 0 means failure (\(a_m(t)\) was “bad for minimization”) and 1 means success (\(a_m(t)\) was “good for minimization”).

In the algorithms presented here, we have made the following choices.
Response. When is the automaton rewarded / punished? I.e. how is $\beta_m(t)$ selected? We use two possibilities: altruistic and selfish mode\(^3\), as follows.

1. In altruistic mode, the $m$-th automaton perceives success when the total inconsistency

   $$G(t) = J(x_1(t), \ldots, x_M(t)) \tag{39}$$

   is decreased, i.e. for all $m \in \{1, 2, \ldots, M\}$ we have

   $$\beta_m(t) = \begin{cases} 
   1 & \text{if } G(t) < G(t-1) \\
   0 & \text{if } G(t) \geq G(t-1). 
   \end{cases} \tag{40}$$

   Note that in this case, at time $t$, every element of $[\beta_1(t), \ldots, \beta_M(t)]$ has the same value, i.e. the automata are rewarded or punished as a team.

2. In selfish mode, the $m$-th automaton attempts to minimize the partial inconsistency $J_m$ associated with its own $m$-th sentence. Define

   $$G_m(t) = J_m(x_1(t), \ldots, x_M(t)). \tag{41}$$

   Then a selfish automaton is succesful when partial inconsistency is decreased, i.e.

   $$\beta_m(t) = \begin{cases} 
   1 & \text{if } G_m(t) < G_m(t-1) \\
   0 & \text{if } G_m(t) \geq G_m(t-1). 
   \end{cases} \tag{42}$$

Action. The actions performed by the $m$-th automaton relate to the choice of the truth value $x_m$. Note that $x_m$ can take values in the uncountably infinite set $[0, 1]$, while the VS LA accomodates a finite set of actions. This problem can be resolved in several ways. One simple-minded approach is to discretize the set $[0, 1]$ into a finite set $\{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K}{K}\}$; some more sophisticated approaches appear in [2, 21, 25, 31]. However we choose a different approach, where each automaton is endowed with three ($K = 3$) possible actions: decrease $x_m$ by $\Delta x$, leave it unchanged, or increase it by $\Delta x$. The next question is: what should the increment $\Delta x$ be? We have experimented with fixed $\Delta x$, but we have obtained better results with a variable $\Delta x$, equal to $G(t)$; in this way smaller steps are taken close to the global minimum 0; we have empirically observed that this improves convergence to the minimizer $(x_1, \ldots, x_M)$. Hence the actions available to the $m$-th automaton at time $t$ are

$$x_m(t) = x_m(t-1) - G(t-1) \text{ w.p. } p_m^1(t-1) \tag{43}$$
$$x_m(t) = x_m(t-1) \text{ w.p. } p_m^0(t-1) \tag{44}$$
$$x_m(t) = x_m(t-1) + G(t-1) \text{ w.p. } p_m^2(t-1) \tag{45}$$

To run (43) we also need to initialize $x_1(0), \ldots, x_M(0)$; these are always chosen randomly from a uniform distribution in $[0, 1]$\(^4\).

Update Mode. We use two variations regarding the sequence with which the automata choose and evaluate actions and update probabilities.

1. Parallel update: at time step $t$ apply (43) (and (40) or (42)) to all $m$ simultaneously.

2. Serial update: at time step $t$ an $m$ (the $m$-th automaton) is selected randomly and (43) is applied for this $m$ only.

For action probability update we use the classical scheme presented in [20]. Namely, with reward rate $a_i$ and punishment rate $a_p$ and for $m \in \{1, 2, \ldots, M\}$, we apply a reward update

$$p_m^i(t) = (1 - a_i) \cdot p_m^i(t-1) + a_i \text{ for } i = k$$

when $\alpha_m(t) = k$ and $\beta_m(t) = 1$, and a punishment update

$$p_m^i(t) = (1 - a_p) \cdot p_m^i(t-1) \text{ for } i = k$$

and $\alpha_m(t) = k$ and $\beta_m(t) = 0$. We always initialize this algorithm with $[p_m^1(0), p_m^2(0), p_m^3(0)] = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$.

4. Experimental Validation

In this section we use numerical experiments to evaluate the four LA algorithms listed in Table 2. The names in the first column denote the style of operation, namely: PA means Parallel/Altruistic, PS means Parallel/Selfish, SA means Serial/Altruistic and SS means Serial/Selfish.

<table>
<thead>
<tr>
<th>Name</th>
<th>Algorithm Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA</td>
<td>Use (43), (39), (40), (45) / parallel update;</td>
</tr>
<tr>
<td>PS</td>
<td>Use (43), (41), (42), (45) / parallel update;</td>
</tr>
<tr>
<td>SA</td>
<td>Use (43), (39), (40), (45) / serial update;</td>
</tr>
<tr>
<td>SS</td>
<td>Use (43), (41), (42), (45) / serial update.</td>
</tr>
</tbody>
</table>

\(^3\)The rationale for the selection of these terms, with their anthropomorphic connotations, is explained in Appendix A.

\(^4\)It is worth remarking that our selection of action set makes our LA an intermediate between finite action learning automata (FALA) and continuous action learning automata (CALA). Strictly speaking, the automaton may take an infinite number of actions, i.e. it may select any $x$-value from the infinite set $[0, 1]$, and this corresponds to a CALA. However, the automaton only maintains a list of three action probabilities (decrease, stay, increase) and hence for practical, computational purposes it is better understood as a FALA.
For the evaluation we use the nine problems listed in Section 2. For each problem we run each of the four algorithms for 100 repetitions; in each repetition the algorithm runs for a maximum of 10000 time steps, but a termination criterion is also applied to stop at an earlier time, if the average inconsistency falls below $\varepsilon = 10^{-5}$ (which practically amounts to finding a consistent truth value assignment). For a complete specification of the algorithms, we must also choose values for the reward and punishment rates $a_r, a_p$. We have chosen (by trial and error) $a_r = a_p = 0.5$ which works well over the entire set of problems examined here. We have found (also by trial and error) that values close to 0.5 give similar results; in other words our algorithms are fairly robust with respect to the $a_r, a_p$ values.

We evaluate the algorithms using two basic quantities. First, the percentage of repetitions (out of the total one hundred) in which a consistent truth value assignment was found is denoted by $c$ and plotted in Figure 1. Second, the average execution time, measured by the logarithm (with base 10) of the number of time steps until termination is denoted by $T$ and plotted in Figure 2 (so in Figure 2 a $T$ value of 2 indicates 100 update steps until termination, 4 indicates the maximum possible number of 10000 steps and so on).

The following remarks can be made.

1. Selfish automata (both parallel and serial) have excellent performance, locating a $c$ value practically equal to 100% in every experiment. Serial altruistic automata also have a $c$ near 100% in almost every experiment, but perform very poorly in experiment 5.1; parallel altruistic automata also perform poorly in experiment 6. Hence selfish automata perform, on the whole, better than altruistic ones.

2. In every case where a high $c$ value is achieved, the corresponding truth value assignments are also very close to true solutions of the truth value equations (i.e. the algorithm not only achieves a low $J$ value, but also a good approximation of an actual solution of the truth value equations).

3. Regarding speed of execution, we see that the LA find good truth value assignments relatively quickly (between 100 and 1000 steps) except for the case of the hard problems 5 and 6.

4. Problem 5, eq.(32), appears to be a hard problem. Using the approximate solutions found by the LA algorithm we have been able to postulate and check that an exact solution is $\bar{x} = [0.95, 0.85, 0.15]$; we are not aware of any other solution and we do not have a way to find one, if it exists. Problem 5, eq.(33) (the same logical system but now implemented with product t-norm and algebraic sum t-conorm) admits at least three distinct solutions, namely

\[ [0.9027, 0.8051, 0.1973], \]
\[ [0.2384, 0.5440, 0.3981], \]
\[ [0.0473, 0.0872, 0.9473]; \]

each of these solutions has been located by at least some of the repeated runs of our LA algorithms.

5. Problem 6, eq.(34) has the exact solution

\[ \bar{x} = [0.875, 0.225, 0.675, 0.8750] \]

(again discovered through the approximate solutions obtained by the LA algorithms); we are not aware of another exact solution. Problem 6, eq.(35) has the solution
\( \tau = [0.9505, 0.2922, 0.5595, 0.7995] \)

It is somewhat surprising that selfishly behaving automata can jointly achieve the result which is best for the entire system, i.e. the minimization of the total inconsistency. It is tempting to draw analogies between this observation and the game-theoretic literature on the emergence of cooperation between selfish agents. This is a topic which has been studied to great extent in the game theory literature (as a small sample we mention the books by Hofstadter [12, Chapters 29-31], Gauthier [10] and Binmore [3] and also the papers [24,32]). Indeed, from the mathematical point of view, a non-zero-sum game is simply a problem of decentralized optimization of (one or more) target functions, all of which depend on the same set of variables, each variable being controlled by a selfish agent. The reader will recognize that this description applies equally well to the problem we have studied in this paper; this point of view is also well presented in the LA literature; see [20,26]. Hence there is an implicit connection between the current work and game theory. We discuss this connection further in Appendix A.

5. Conclusion

In this paper we have introduced a theoretical framework, based on fuzzy logic, to reduce logical self-reference and paradox to the problem of solving systems of numerical equations. Then we have presented an LA-based computational approach to solve the numerical problem, which amounts to resolving a wide class of self-referential paradoxes (the Liar paradox is the most celebrated member of the class).

We plan to further research the following related issues. First, we want to compare our approach to the use of “standard” variable structure LA with a finite number of actions (this case can be applied to fuzzy logics with a finite number of truth values and in conjunction with discretization of the continuous set [0,1]). Second, we intend to analyze theoretically the LA algorithms presented in the current paper, with special emphasis on expediency, \( \varepsilon \)-optimality, convergence rate etc. Third, we want to adapt our approach to the theory of explanatory coherence [28,29] which, as far as we know, has not been applied to self-referential sentences. The connection is natural: since self-referential sentences make claims about each other’s truth values, it is rather straightforward to setup a network with one node per sentence and with connections which can be either reinforcing or inhibiting (depending on what sentences say about each other).

A. Appendix: Learning Automata, Game Theory and Informal Reasoning

In Section 4 we have mentioned that (from the mathematical point of view) a non-zero-sum game is simply a problem of decentralized optimization. This point of view illuminates the connection between the resolution of self-referential systems and a team of learning automata playing a game. Of course this is an “anthropomorphic” view.

What entity is represented by each automaton participating in this “reasoning game”? In this Appendix we offer a possible answer which has motivated us to use learning automata for the resolution of self-referential systems.

We introduce the following hypothesis: when a reasoner is presented with a self-referential system, the self-referential sentences compete for acceptance by the reasoner. This hypothesis is a variation of an idea presented by R. Dawkins [7] and popularized by D. Hofstadter [12]. Dawkins defines a meme to be a unit of cultural transmission, or a unit of imitation ... Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or building arches. Just as genes propagate in the gene pool via a process which, in the broad sense can be called imitation ... if the idea catches on, it can be said to propagate itself, spreading from brain to brain. [7]

Dawkins and Hofstadter suggest that meme competition evolves over many thinkers and many generations of thinkers, resulting in a process of evolutionary selection. In the current paper, however, we consider a single thinker reasoning about the validity of several self-referential propositions. Each proposition “attempts” to ascertain itself in the thinker’s mind by adjusting its truth value to minimize the inconsistency between its assigned truth value and the truth value computed as a function of the truth values of the remaining propositions; the “best policy” for a sentence is to adopt the truth value which minimizes its inconsistency. We repeat that this is a highly anthropomorphic interpretation of the situation; under this interpretation the algorithms of Section 3 can be seen as models of human reasoning as a truth value adjustment process, with one automaton corresponding to each sentence/meme.

The above interpretation can be used as a metaphor to motivate our use of LA, but is it a psychologically plausible model of human reasoning about self-referential propositions? It certainly does not model reasoning by the rules of formal logic; but in the psychology literature it has often been reported that humans do not always reason by the rules of formal logic [6,14,22,23]. It has also been argued that belief change consists in “propositional update”; for example, [8] re-
ports that, when a human is faced with a collection of contradictory propositions, she will reject some of these propositions (i.e. assign to them truth value equal to zero)\textsuperscript{5}. If fuzzy logic and the use of continuous truth values are acceptable, partially believing a proposition (i.e. assigning to it a truth value in the interior of $[0, 1]$) is an alternative to outright rejection.

Is it plausible that human reasoning is implemented by a dynamical system of the form (45)? Generally humans do not consciously reason by updating truth values\textsuperscript{6}. On the other hand, the hypothesis that reasoning at the subcognitive level corresponds to the evolution of a dynamical system is one of the fundamental ideas of connectionism [19] and has also been used in other contexts [13]. And the idea that “practical” reasoning involves constraint satisfaction has been expressed eloquently by P. Thagard and his collaborators [29] in the form of coherence theory [28,30]. Hence the operation of a learning algorithm similar to (45) at the subcognitive level is not implausible.

Given the above considerations, a game theoretic approach to the problem of (human) reasoning is not out of place. It is implicit in the LA literature and has also been expressed in several other contexts, most notably in the literature of market-driven artificial intelligence [9,27], but also in [1], [5] etc.

We repeat that in this Appendix we have presented a hypothetical interpretation of the mathematical methodology presented in the main body of the paper. The success of our methodology in computing consistent truth values, is a fact, not a matter of interpretation; on the other hand, its plausibility as a realistic model of human reasoning is an open question.

References


\textsuperscript{5}This is also the underlying theme of nonmonotonic inference[4].

\textsuperscript{6}They do so occasionally, for instance the reasoning by which the Liar paradox is elicited (see Section I) is one such iterative procedure – see also [11].