

Automating the Abductive Inference Loop

Coen J.C. Stevens, Bart-Floris Visscher and Thomas R. Addis

Abstract—We present a novel approach to handle uncertainty, by assuming an irrational view of the world thus breaking with the tradition of taking a rational view. Changing this view allows us to think differently about the nature of uncertainty. Based on this new understanding, we propose a system that can cope with an imprecise, uncertain and changing ‘world’. The proposed solution handles irrationalities, which includes uncertainty, by generating different hypotheses of the environment and testing these for their validity thereby fully automating the abductive loop. Since the proposed system makes no assumption that any knowledge about the world will ever be completely reliable, all knowledge is susceptible to change. With the full automation of hypothesis generation, inference mechanism and validation, there is no apparent problem domain with which the proposed system could not cope.

I. INTRODUCTION

The success of AI is undeniable, especially when we look at today’s computer games where chess playing programs beat Grandmasters and the AI opponents in other strategy games become more and more believable in showing convincing, life-like behavior [7]. Despite these impressive results, there have also been a number of significant problems which AI research has run into. One fundamental problem is what H. Dreyfus calls the ‘Commonsense Knowledge Problem’ [6], which is defined by three subproblems: How everyday knowledge must be organized so that one can make inferences from it; How skills or knowledge can be represented as knowing-that; and How relevant knowledge can be brought to bear in particular situations.

According to Dreyfus, common-sense understanding requires a notion of relevance, characterized by context, not compatible with atomistic, symbolic data structures and discrete computations. What is relevant in one setting might not be so in another, depending on the purpose, resulting in that no static set of rules can ever describe knowledge holding truth for all settings. H. Collins, another critic of AI, used this same difficulty to argue the impossibility of simulating human intelligence by the methods of traditional AI [5]. So even while technology pushes us forward, giving us machines that perform mathematically definable operations millions of times faster than any human and execute them with almost perfect accuracy, we keep failing in the creation of human intelligence.

C. Stevens is with the Man-Machine Interaction group, University of Delft, 2628 CD Delft, The Netherlands, In association with the Netherlands Defence Academy (NLDA) j.c.stevens@ewi.tudelft.nl

B. Visscher is with Clarity Support Limited bart-floris.visscher@clarity-support.com

T. Addis is with the School of Computing, University of Portsmouth, PO1 3AE Portsmouth, UK tom.addis@port.ac.uk

Acknowledging the Commonsense Knowledge Problem shows what makes modeling intelligence for human interaction and language so different than for (say) AI in chess. One may hypothesize that the answer lies in the difference in environments. In non-simulated environments we need to deal with an imprecise, uncertain and changing world about which it is hard to be categorical [14], while in a simulated and well-defined environment (like a chess computer game) we are bounded by a fixed set of rules. Many different formal techniques have been developed over the past decades for dealing with uncertain information [8]. All of them try to rationalize the environment in their own way, but according to Addis et al. [2] they skip the ‘irrational’ nature of the environment in doing so. This irrational nature is characterized by the notion of the ubiquitous existence of ‘irrational sets’ (see Section II) that emerge from a continually changing knowledge of the world. Dealing with these irrational sets is acknowledged as a Grand Challenge in computing [4][9]. Addis et al. propose a new paradigm for coping with the world by shifting from a ‘rational’ to an ‘irrational’ view and to increase the capability of tracking changes. In order to make sense of a dynamic and uncertain world, we can not assume any static knowledge about the world and we therefore need a mechanism to handle the dynamic knowledge in the form of irrational sets.

In this paper we present a novel approach in handling uncertainty. We propose a new mechanism for dealing with a changing environment using the notion of irrational sets. The mechanism handles irrationalities by generating different hypotheses of the environment and testing these for their validity thereby fully automating the abductive loop.

The paper is organized as follows. In the next section, we explain the notion of irrational sets and the irrational view after which we show the need for abductive inference in tracking irrational sets. In Section 3, we present our mechanism, the automation of the abductive inference loop, and show that it conforms to the irrational view.

II. TRACKING IRRATIONAL SETS

A. Irrational sets

In explaining how we can track irrational sets, we first provide the definition of a rational and irrational set.

A ‘Rational set’ is a set where there is a finite set of rules that can include unambiguously any member of that set and unambiguously excludes any non-member of that set [4].

In cases where it is not possible to construct a finite set of unambiguous rules, the set is defined as an ‘irrational set’,

which means that an irrational set is a type of set where memberships may change in an unpredictable way. The name rational and irrational set comes from a similarity between rational and irrational numbers. Like the relationship between some rational and irrational numbers, a rational set can only approximate some irrational sets but there will always be some error [12].

An 'Irrational set' is a set where no finite set of rules can be constructed that can include unambiguously any member of that set and, at the same time, unambiguously exclude any non-member of that set [4].

Wittgenstein [13] already found years ago that the normal use of language is riddled with example concepts that cannot be bounded by logical statements that depend upon a pure notion of referential objects. One of Wittgenstein's examples is an attempt to define a 'game', which is a good example of an irrational set. When assuming that games define a rational set, one might try to find the rules that define the boundary, but to this date, no rules have been found that can exclude all non-members and include all members. If for no other reason that people cannot agree where the boundary lies.

Since computer work is based on mathematically defined operations, computers can only handle rational sets. How then can a mechanism be built to approximate irrational sets? A summary of the rational and irrational set characteristics (See Table I) by B. Visscher [12] shows that we need a radically different mechanism.

TABLE I
RATIONAL AND IRRATIONAL SET CHARACTERISTICS

| Set features | Rational set | Irrational set |
|--------------|--------------|-----------------------------------|
| Boundary | Static | Dynamic |
| #Boundaries | Single | Multiple |
| Consistent | Always | Not necessarily |
| Membership | Clear | Can be ambiguous or contradictory |

Based on these characteristics we argue that for tracking irrational sets, a system needs to have the following three characteristics [12]:

- 1) Dependent on time. As shifts occur over time and cannot be predicted, the shifts would be reflected in the order.
- 2) The learning period never ends. The system must continue evaluating and learning when presented with new ideas and situations.
- 3) Showing some form of randomness. Since we can not make the assumption that any knowledge about the world will ever be completely reliable, all knowledge is susceptible to change. To allow for exploration of this we need to keep all options open.

B. The Irrational view

The rational view, assumes that everything can be described in terms of rational sets, which would only appear

irrational merely because at the moment we are dealing with incomplete and therefore imprecise information [12]. With enough additional information all conflicts can be resolved. Irrationality is considered subordinate to rationality and considered a temporary state of affairs that will be solved. The new paradigm captured in the irrational view, rejects this assumption of complete information not because the environment itself may be irrational but because many systems within the environment are not complete. In order to cope with these, the system itself has to cope with irrationalities. As such, the irrational view assumes that not everything can ever be potentially unambiguously described, but know that this approach is exactly what we are doing when we are trying to handle uncertainty within the rational view. Working within the irrational view is more general as it encompasses rational environments, but also irrational environments. Everything we can do with a rational set is also included in irrational sets, but not visa versa.

An example of an irrational environment is human interaction. Here we consider humans to be incomplete systems, because people are limited in the amount of data they can collect and just can not see the world completely rationally as people can not know everything. The world appears to be irrational to them and they just try to deal with the irrationalities they encounter. In interacting with humans the system needs to deal with human views and conceptualizations.

The irrational view is in a real sense not new nor does it present a solution to the problems AI research run into, it only suggest an acknowledgement that there is a formal analysis that can take account of irrational sets. The consequences are nonetheless considerable, as we will show that most current AI techniques do not conform to the irrational view and are therefore inadequate in dealing with and indefinite world.

The following techniques aim to use (adaptive) learning, or evolutionary computation to create programs that are, in some sense, intelligent:

- Fuzzy Systems
- Probabilistic Reasoning, Bayesian Networks
- Artificial Neural Networks
- Evolutionary Computation

With Fuzzy Systems the 'fuzzy sets' are often mistaken for irrational sets, but one should be reminded that fuzzy sets remain in the rational domain. Fuzzy sets are rational in that members are assigned an explicit, ordinal membership number. We can extend the classical set of chair membership where Chair=1 and not-Chair=0 by assigning intermediate values (e.g. 0.5), but such assignments can be expressed by a finite set of rules and therefore it is within the domain of rational sets. Similarly for Probabilistic Reasoning, where a probabilistic assignment of membership is also rational because it is specified by a rule in the form of a ratio of integers.

We mentioned that in order to keep track of irrational sets the learning period must never end and therefore the system must continue evaluating and learning for ever. Neural

Networks, which have a fixed learning period are therefore inappropriate to deal with the problem of shifting sets. Using a Neural network to track changes will eventually result in ‘overtraining’ of the network. Changing sets would only appear as noise to the training set and result in conflicts and ambiguity. Evolutionary algorithms on the other hand are capable of adapting forever, as their solution may constantly evolve.

C. Scale of irrationality

Taken from the irrational view that the world exists out of irrational and rational sets, does not mean that something is either completely rational or completely irrational. It is important to notice that there is a scale of irrationality, which all depends on how dynamic the environment is; how fast changes occur. Irrationality can go from unity, where no distinctions are made, to rational where the set distinctions can be made perfectly without any false-positives, false-negatives, ambiguity or contradictions (see Figure 1).

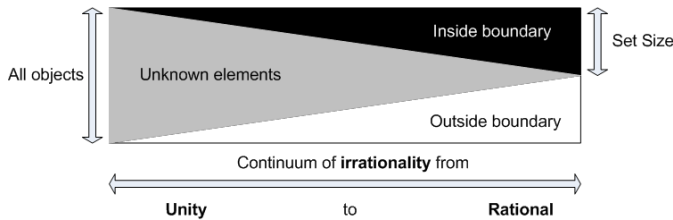


Fig. 1. Scale of irrationality

In cases where the environment acts ‘more or less’ rationally, we would not abandon the rational view. It is probably less efficient in terms of computing to model rational sets as being irrational than otherwise. So it is important to have a measure for irrationality to assess what kind of model to use. This may be defined as the degree of contextualization needed to rationalize [12].

In situations where ‘irrationality of a set’ is undetermined, it is safer to assume we are dealing with an irrational set, because irrational sets include rational sets. Although it might be less optimal in terms of computational speed, we would have a working solution that continues to work. If we were to assume a rational set while modeling an irrational one, we expect performance degradation and eventual failure [12].

D. Abductive Inference Loop

The introduction of irrational sets has consequences for the inference mechanism. In the case of a formal system based upon rational sets deductive inference depends upon the preservation of truth, which is purely syntactic as long as the coherence of the rational set is kept at all times. Irrational sets destroy the coherence of the sets, which means that we can no longer rely upon the deductive inference mechanism. Addis et al. proposed to use the abductive inference loop (see Figure 2), which was originally proposed by C.S. Peirce [11] and modeled by Addis and Gooding [3].

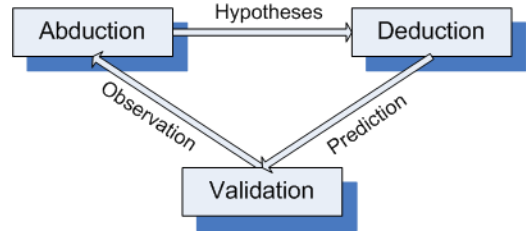


Fig. 2. Abductive inference loop

Without deduction and the preservation of truth we have to cope by some other means. To this end truth is replaced by belief and deduction by experiment and validation. The system believes in a set of hypotheses which make up the formal part of the system used for deductive reasoning. In this way we can make predictions about the world. These predictions can be validated by observations in the world. Validation provides a mechanism through which conclusions can be justified. The process of validation adjusts the beliefs and this means we adjust the formal model. What is important to notice here is the need for purpose. Without purpose there is no criteria for success and therefore no mechanism for validation [1].

III. AUTOMATING THE ABDUCTIVE LOOP

A. Proposed mechanism

The working model for the abductive loop by Addis and Gooding [3] can be used to great extent in automating the abductive inference loop. We use the belief profile model, which defines a range of beliefs about the world. A belief is represented as the combination of a hypothesis about the world and a number between zero and one that reflects the subjective probability that the hypothesis is ‘true’ and can be assessed by observing the system’s actions [3]. Each hypothesis represents a model of the behavior of the world that we want to capture, which could be either rational or irrational. Different hypotheses may hold different models with which the system tries to relate to the world. Depending on what is being modeled and what the system’s purpose is, different types of hypotheses are needed. For instance a system for the evaluation of cocktail drinks will need different type of hypotheses than for facial recognition. In modeling the hypotheses, we will present a general structure for the hypotheses, which makes it compatible with any type of purpose.

Despite having a mechanism for learning as belief revision, which is conforming to the irrational view, the model of Addis and Gooding is not able to track changes in the world that lie outside a predefined set of possibilities. The irrational view tells us the world is indefinite and therefore there can never be a fixed set of rules or hypotheses for describing the world, unlike Addis and Gooding’s model, which can only work with a fixed set of hypotheses. In order to track changes in the world we need to have a mechanism that can change the hypotheses and generate new ones (see Figure 3).

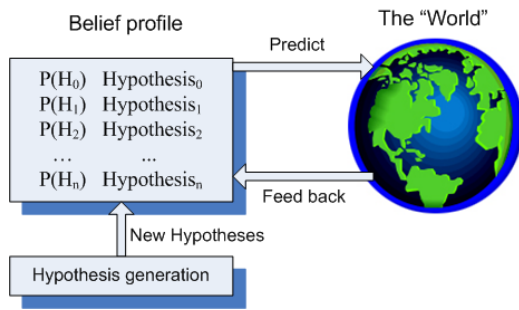


Fig. 3. Belief system with hypothesis generation

The generation of hypotheses is a key element in modeling irrational behavior as the system can introduce new concepts and shift boundaries thereby continuously adapting to its environment. In the rational view every case would have to be pre-designed or pre-determined, something which is impossible to do as the system should therefore account for all the possible future situations. Concepts can change under time and social context.

Before explaining the mechanism for learning as belief revision and the ‘genetic’ mechanism for hypotheses generation in detail, we define the model for the basic elements in the systems; the hypotheses.

B. Modeling hypotheses

We propose to model a hypothesis as a logically defined network. The network consists of input, output and intermediate nodes, which can be set active(1) or inactive(0). In the network we define two types of relations (connections) between nodes, which both are many-to-many relationships:

- Stimulator: A node can stimulate another node. If the first node is activated, then it will stimulate the second node and thereby activating it.
- Inhibiter: A node can inhibit another node. If the first node is activated, then it will inhibit the second node and thereby setting it inactive.

The next figure shows an example of a network with four input nodes ($i_{0,1,2,3}$), two intermediate nodes ($c_{0,1}$) and three output nodes ($o_{0,1,2}$).



Fig. 4. Hypothesis Network, where \rightarrow and $-|$ respectively stand for stimulates and inhibits

The network is only representing a function and therefore can not learn or adapt itself. This in contrary to an Artificial Neural Network (ANN) with McCulloch-Pitts neurons, which can be trained to adapt its weights and thereby its function. Still our network is not a regular input-output mapping function as the network could also give its own

feedback (see Figure 4, where output o_0 stimulates input i_3). Every node can stimulate or inhibit any other node, including itself, and because there is no sequential ordering in the stimulating and inhibiting of nodes, we need to compute in parallel. In this way we can have a never ending progress through the network after activation of some of the input nodes, therefore we need to limit the number of steps through the network or set a time limit. When we reach the limit we count the number of times each output concept has been activated, which we normalize over all outputs, resulting in a probability distribution over the output concepts.

The network is capable of representing any logic function. Every function is expressible in terms of AND, OR and NOT functions or in terms of NAND (or NOR) alone as by use of DeMorgan’s theorem, an AND can be turned into an OR by inverting the sense of the logic at its inputs and outputs. For the creation of a NAND, we need a network that describes the behavior in table II.

TABLE II
NAND FOR SIGNALS A AND B, AND OUTPUT Y

| A | B | Y |
|---|---|---|
| F | F | T |
| F | T | T |
| T | F | T |
| T | T | F |

To represent one Boolean signal, two signals are needed. This means to get signals representing the input values for a NAND gate, we need a total of four input signals to represent the cases TRUE for $A(T_a)$, FALSE for $A(F_a)$, TRUE for $B(T_b)$ and FALSE for $B(F_b)$. For the output only one signal Y is used and we therefore need two signals here representing the TRUE and FALSE case for Y , which respectively are T_y and F_y . The next Table represents the NAND behavior with these six signals, instead of the three in previous Table II.

TABLE III
NAND WITH SINGLE CONCEPT REPRESENTATION

| T_a | F_a | T_b | F_b | T_y | F_y |
|-------|-------|-------|-------|-------|-------|
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |

One possible network implementation of the NAND gate is presented in the next Figure.

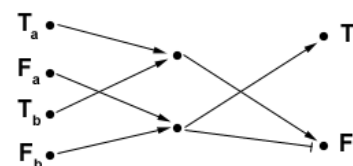


Fig. 5. NAND network

With the type of network we explained here, we can create any function thereby having hypotheses that can model any type of behavior. For instance a hypothesis can represent the same behavior as a trained Artificial Neural network, which is basically an input-output mapping function.

C. Hypotheses generation

To generate potentially ‘better’ hypotheses, which closer fit the irrational behavior of what is being modeled and adapt to unpredicted changes, we propose an evolutionary algorithm. The algorithm should evolve the solutions by breeding offspring and creating new populations of hypotheses. By using biological inspired methods of crossover and mutation, we want to breed new hypotheses out of (initial) existing ones. In order to breed with networks we first present how to define the networks in terms of genes.

Since every active concept that comes from either an input or from within the network can only stimulate or inhibit other concepts, the whole network can therefore be described in the following IF-THEN terms:

IF ‘concept’=1 THEN Stimulate OR Inhibit ‘concept’

We propose a pool of genes wherein gene strings define these IF-THEN terms. Figure 6 presents an optional mapping from genes to the network described in previous Figure 4.

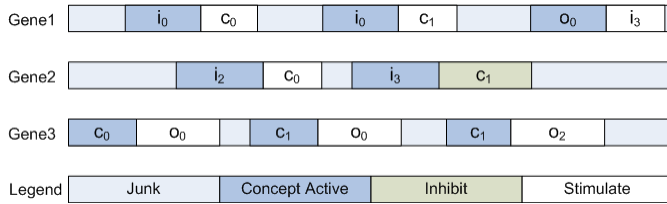


Fig. 6. Example of genes defining a network

The genes are composed of valid IF-THEN term combinations and ‘junk’, where junk is what fills up the genes with sequences for which no function has been identified. The reason for introducing junk is inspired by what can be seen in nature, referred to by ‘Junk DNA’. Junk could give the following advantages:

- Much junk gives point mutation a smaller change to have any effect. In this way junk acts as a protective buffer against genetic damage (errors) and negative point mutations.
- Junk could provide a reservoir of sequences from which potentially advantageous new IF-THEN terms and concepts can emerge.
- Functions switched off by mutation, remain present in the genes and could be switched on again by future mutations. In this way we can have dormant gene parts.

It is important to notice that, no matter what kind of crossover or mutation, we always end up with genes representing a valid network as each possible network can always be processed. Crossover and mutation may not only affect the relations between concepts, but may also create new

concepts by mutating concepts or emerge them from junk. The creation of new concepts within the hypotheses is a key element in modeling irrational behavior as it allows the system to adapt to new situations that lie outside a predefined set of possibilities. Whether a new network, with possible new relationships and concepts, is an improvement over the other hypotheses is for the belief system to find out.

D. Gene Coding

One way to implement the genes is to create genetic strands of bit strings. In this way a gene is represented by a string filled with zeros and ones. The bit string can be read from left to right in order to find the encoded π 's between the junk. To explain how this works let us start with the following example presented in figure 7.

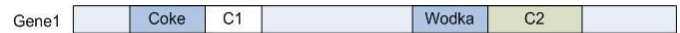


Fig. 7. Gene Example

If we want to identify valid π 's along the bit string, then we need the use of bit code Identifiers. Take for example identifier ‘00000’, whenever we find bit string ‘00000’ inside the gene it means that it is followed by a valid π . In our example this means that the two π 's in the string are preceded by ‘00000’.

‘00000’ ‘Coke’+‘C1’ and ‘00000’ ‘Wodka’-‘C2’

The next step is where we need each concept to be represented by a unique bit code. When we build the genes for the first time, then we start with the list of known concepts which are present in the network and Δ program (input ,output and intermediate concepts). But in time we want new concepts to emerge from junk or by concept mutation. These emerged concepts should have their own unique bit code. Paragraph III-E describes the mechanism for the needed progressive bit state coding. For the example we could have the following unique bit codes:

‘00000’ ‘0101’+‘1000’ and ‘00000’ ‘1100’-‘1110’

Because there are only two types of π 's (+ and -) we can replace the + (stimulator) by a 1 and the - (inhibitor) by a 0.

‘00000’ ‘0101’ 1 ‘1000’ and ‘00000’ ‘1100’ 0 ‘1110’

With all this every part of the Delta-Program can now be mapped onto a string of bits. The only thing missing now is Junk. Junk will be represented by a sequence of random numbers of zeros and ones. Junk cannot contain the ‘00000’ sequence, because this infers that we have found an identifier after which follows a valid π . Adding some junk to our example results in the following possible sequence of bits for Gene 1. (The | is added at some places to increase readability, where they separate junk, identifiers and patterns)

0100110101|00000|0101|1|1000|11010101010|00000|1100|0|1110|0|

It is not necessary for a Δ program to be mapped onto only one gene, the valid π 's may be spread over multiple genes. For now there are no guidelines or rules of thumb for the

number of genes, but the genes themselves should consist for about 97 percent out of junk. Genes don't have a fixed length as the amount of junk and the number and length of π 's may vary. The fact that also the length of concepts may vary is due to the used concept coding which is discussed in the next paragraph.

E. Flexible bit state coding

As stated earlier, each concept needs to be represented by a unique bit code to distinguish between concepts. This would have been easy if the list with concepts was a fixed set, but in time we want new concepts to emerge from junk or by concept mutation. The reason why we want new concepts is because they could be potentially advantageous in forming the best evaluation function. Only there isn't really an ultimate everlasting evaluation function to be found (or approximated), due to the irrational nature of it. People will change their opinion on taste in time and context. So we need to be able to form new concepts to track the user's taste.

Another point is being dynamical. For instance when we would like to add new concepts (ingredients) to our input space we don't want to encode the genes all over again. Introducing new concepts asks for flexible coding.

What we don't want is to start by stating a fixed number of bits to represent the concepts, say for example a number of 8, because then we always have a limit to the number of concepts, in this case 8bits = 256 states. We don't want this fixed limit, when we reach the limit we want to expand the number of bits representing states. If we would state a fixed number of bits, then we are limited in our flexibility. Flexible bit state coding means that the length of concepts may vary. This makes it a bit more difficult when we start reading the gene bit string from left to right decoding the concepts.

Reading the gene bit string from left to right we should be able to find the π 's not knowing before hand which concepts we will find. The only thing we can find in the first place is the π identifier. We start looking for the first identifier sequence, which is "00000" in the next gene bit string example.

0100110101|00000|0010100010101110101010000010

Once we have found the identifier sequence we know that a valid π will follow, defining one concept that will inhibit or stimulate another concept. Because we don't know the length (number of bits) of the concepts following the identifier, this needs to become clear from the bit pattern.

One way to do this is to define the length of the concept by the number of zeros at the beginning of the bit pattern. In practise it works as follows. When we start reading the gene bit string after the identifier, we count the number of following up zeros. As long as the length of the zero bit pattern is less than the Z-Pattern the length is defined by the Depth, where the Z-Pattern and Depth are arbitrary initialized at 2 and 4 respectively. When the zero bit pattern equals the Z-Pattern, both the Z-Pattern and Depth are doubled. The

concept length (Depth) is thereby defined with the following rules:

Z-Pattern = 2
 Depth = 4
 IF number of zeros < Z-Pattern THEN Depth
 ELSE Z-Pattern = 2 * Z-Pattern AND Depth = 2 * Depth

Using these rules will result in what is shown by table IV, the Concept lengths for some of the Zero bit pattern lengths.

TABLE IV
 ZERO BIT PATTERN LENGTH AND CONCEPT LENGTH RELATION

| Zero bit pattern length | Concept length |
|-------------------------|----------------|
| 0-1 | 4 |
| 2-3 | 8 |
| 4-7 | 16 |
| 8-15 | 32 |
| 16-31 | 64 |
| ... | ... |

The previous rules are not the one and only way to implement this flexible behaviour. Both the initialization of the Z-Pattern and Depth is arbitrary and could have been any positive value as long as the Depth > Z-Pattern. Also doubling the Z-Pattern and Depth is not the only possibility to increase the limits.

But in any case flexible coding in this way means that not all the possible bit combinations are used. For instance out of the 16 possible states with 4 bits, only 12 states are valid as shown in figure 8.

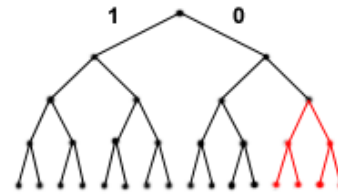


Fig. 8. Bit codes length 4

The states '0011', '0010', '0001' and '0000' aren't valid as their zero bit pattern defines a different length.

What goes for the length of 4 bits goes for the other lengths as well. The next figure gives an illustration of the part of the bit pattern tree that will be used.

Using only these parts of the tree results in the following number of possible valid states presented in table V.

TABLE V
 NUMBER OF STATES

| Number of bits | Number of states |
|----------------|--------------------|
| 4 | 12 |
| 8 | 48 |
| 16 | 3.840 |
| 32 | 16.711.680 |
| 64 | > 10 ¹⁵ |
| ... | ... |

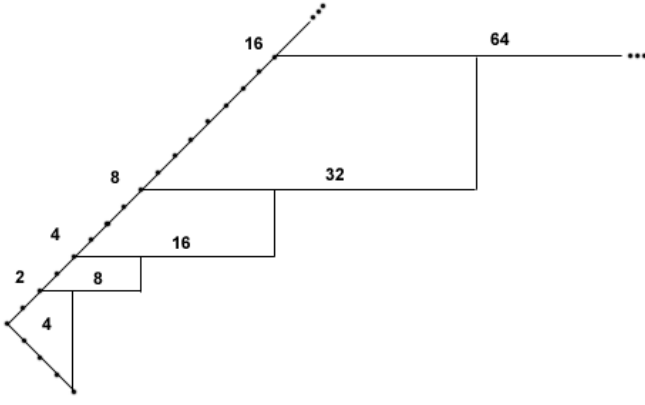


Fig. 9. Flexible bit coding

Looking back at the gene bit string example we should now be able to read out the π 's. After the '00000' identifier we remain with the next sequence of bits.

0010100010101110101010000010

Before we find the first 1 in the string, we already count 2 zeros which define a length of 8 bits (See table IV). So the first concept bit code will be '00101000'. The next bit defines the relation, which is in this case a 1 for stimulating. The final step is to read out the second concept and because its head is only one zero we know it will have length 4 given us a sequence of '0101'.

00101000|1|0101|110101010000010 \longrightarrow
00101000 stimulates 0101

In the remaining string of bits we can find another identifier. When we start reading after this identifier we run out of bits.

11010101|**00000**|10...

In this case we break off and consider the end of the gene to be junk. For the next gene we start all over again by searching for an identifier.

F. Belief system

The final ingredient for the automation of the abductive inference loop is the mechanism for learning as belief revision. Depending on the data retrieved from interacting with the world or a given data set, the mechanism will adjust the hypotheses's beliefs in order to find the best hypotheses. The data is used for an experimental setup thereby represented as a table of real numbers that indicate the probability of a result occurring, given that a hypothesis is true. By making experiments that are expected to bear directly on one or more of the hypotheses, we find confirmatory or disconfirmatory evidence which is used to adjust the beliefs. The model for belief revision is presented in the next Figure.

To make belief adjustments we need to determine the expected hypothesis with the following equation:

$$E_{n-1}(H/R_e) = \frac{E_{n-1}(R_e/H)}{E_{n-1}(R_e)} \quad (1)$$

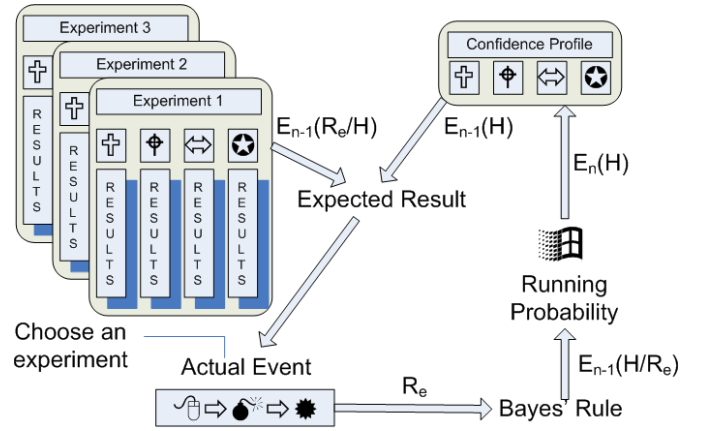


Fig. 10. Belief system

Here R_e is the result of an experiment e and H is a hypothesis. The expected result $E_{n-1}(R_e/H)$ for any experiment will depend upon the belief of each hypothesis and the probability of a result for the hypothesis supposing it is true. To get the probability of a result for a hypothesis means evaluating the experiment.

$E_{n-1}(R_e)$ is the expected result for all hypotheses. Defined by:

$$E_{n-1}(R_e) = \sum_H E_{n-1}(H) * E_{n-1}(R_e/H) \quad (2)$$

Given a result from experiment e the belief probabilities (confidences) for each hypothesis, can be modified to $E_n(H)$ by adapting the above equation to the following:

$$E_n(H) = \frac{(N-1)E_{n-1}(H) + E_{n-1}(H/R_e)}{N} \quad (3)$$

Flexibility is defined as $\frac{1}{N}$ and reflects responsiveness to new results. The Flexibility value creates a time window for belief in previous evidence. Flexibility is necessary for handling irrational sets, because it reduces the importance of past evidence. When the world is changing rapidly, then 'old' evidence may no longer apply for the new situation.

For more details on the belief system we refer to the paper by Addis and Gooding [3].

IV. CONCLUSION

We presented, with the full automation of the abductive inference loop, a mechanism that can cope with an imprecise, uncertain and changing world. The mechanism conforms to the irrational view and offers a solution to AI problems where traditional AI solutions fail. The failure of traditional solutions can be explained by the hidden assumption of the rational view within them and the irrational nature of the problem. Within the rational view we are not able to track changes in the world that lie outside the predefined set of possibilities. Whether or not we should approach a problem domain using the irrational view, depends on how dynamic the environment is; how fast changes occur. Irrationality can go from unity, where no distinctions can be made, to rational

where the set distinctions can be made perfectly forever without any false-positives, false-negatives, ambiguity or contradictions. The more the situation tends towards unity on the irrational scale, the more uncertain the system will be and the more flexible it needs to become. Since our system assumes no static knowledge about the environment we have achieved the maximum amount of flexibility. Experimental results, of an implementation of the automated abduction loop for the evaluation of ‘cocktail’ drinks where hypotheses model the subjective and changing taste of users [10], are promising.

REFERENCES

- [1] T. R. Addis. Stone Soup: Identifying Intelligence through Construction. *Kybernetics*, 29:849–870, 2000.
- [2] T. R. Addis, J. T. Addis, D. Billinge, D. Gooding, and B.-F. Visscher. The Abductive Loop: Tracking Irrational Sets. In *Proceedings of Model-Based Reasoning in Science and Engineering, MBR’04*, 2004.
- [3] T. R. Addis and D. Gooding. Learning as collective Belief-Revision: Simulating reasoning about Disparate Phenomena. In *Proceedings: AISB99 Symposium on Scientific Creativity*, pages 19–28, Edinburgh, 1999.
- [4] T. R. Addis, B. Visscher, D. Billinge, and D. Gooding. Socially Sensitive Computing, A Necessary Paradigm Shift for Computer Science. In *Grand Challenges in Computing*, pages 1–6, UK CPHC, Newcastle, 2004.
- [5] H. Collins. *Artificial Experts: Social Knowledge and Intelligent Machines*. MIT Press, 1990.
- [6] H. Dreyfus. *What Computers Still Can’t Do: A critique of artificial reason*. 1992.
- [7] D. Livingstone. Turings Test and Believable AI in Games. *ACM Computers in entertainment (CIE)*, 4(1), 2006.
- [8] S. Parsons and A. Hunter. A review of uncertainty handling formalisms. *Applications of Uncertainty Formalisms, Lecture Notes in Artificial Intelligence, Springer*, 1455:8–37, 1998.
- [9] S. Stepney, S. L. Braunstein, J. A. Clark, A. Tyrrell, A. Adamatzky, R. E. Smith, T. R. Addis, C. Johnson, J. Timmis, P. Welch, R. Milner, and D. Partridge. Journeys in Non-Classical Computing II: A Grand Challenge for Computing Research. *International Journal of Parallel, Emergent and Distributed Systems*, 21(ISSN 1744-5760):97–125, April 2006.
- [10] C. Stevens. A Genetic Irrational Belief System. Master’s thesis, University of Delft, 2005.
- [11] R. Tursman. *Peirce’s Theory of Scientific Discovery: A System of Logic Conceived as Semiotic*. Indiana University Press, 1987.
- [12] B. Visscher. *Exploring Complexity in Software Systems. From an irrational model of software evolution to a theory of psychological complexity and cognitive limitations based on empirical evidence*. PhD thesis, University of Portsmouth, 2005.
- [13] L. Wittgenstein. *Philosophical investigations, London, Macmillan*. Macmillan, London, 1966.
- [14] L. A. Zadeh. Soft computing and fuzzy logic. *IEEE Software*, 11(6):48–56, Nov. 1994.