Conscious Machines and Consciousness Oriented Programming

Norbert Bátfai

University of Debrecen

Department of Information Technology

`batfai.norbert@inf.unideb.hu`

August 16, 2011

Abstract

In this paper, we investigate the following question: how could you write such computer programs that can work like conscious beings? The motivation behind this question is that we want to create such applications that can see the future. The aim of this paper is to provide an overall conceptual framework for this new approach to machine consciousness. So we introduce a new programming paradigm called Consciousness Oriented Programming (COP).

Keywords: *programming paradigm, machine consciousness, conscious computer programs, intuitive computer programs, quasi-intuitive Turing machines, ConsciousJ programming language.*

Contents

List of Figures

List of Tables

Listings

1 Introduction

Eugen Wigner wrote in his essay [Wigner, 1967] that *"observation of infants where we may be able to sense the progress of the awakening of consciousness"* is a possible method to solve the mind-body problem. I have three

children and I have been observing them when I can. They are now 3 and 5 years old. The older child has already been perfectly able to arrange the everyday events in time, the younger two haven't been able to speak about it with any degree of accuracy yet. With hindsight, moreover, at the age of 2, they couldn't handle the term timeliness.

In the course of human cognition, there has been a need to know the future from time immemorial. The success of this effort has been culminating at Newton's mechanical world-view in the late 19th century. But since then the quantum mechanics has turned this deterministic world-view upside down, opening the way to use new quantum phenomena of a deeper level of reality. But even though the Orch OR [Penrose and Hameroff, 1998] model of quantum consciousness is an exciting and promising theory, we have to restrict ourselves to investigate computer programs and machine consciousness because computers of nowadays have no quantum computing parts.

We believe that one of the drivers of evolutionary evolving of natural intelligence was the process of replacing, by natural selection, the automatic response of living matter with foresight.

In this work, in compliance with this outlined motivation, we emphasize the pursuit of predicting the future as the cornerstone of the definition of machine consciousness.

1.1 Previous and Similar Works

Several recent studies have included definitions of machine consciousness. For example, [Starzyk and Prasad, 2011] outlined an architectural model inspired by the functional organization of the human brain. Their definition [Starzyk and Prasad, 2011, pp. 9] says that a machine is conscious if the functional components concerned are present at the machine in question. This and similar (for example, CogAff [Sloman and Chrisley, 2002], Lida of GW [Baars and Franklin, 2009]) models typically involve a detailed description of a sophisticated architectural system and focus on the question of "How".

By contrast, in our opinion, the conditions of the definition should be in a format that the fulfillment of these can be easily checked. Accordingly, in this paper we are not interested in the question of "How". We place only one aspect at the heart of the definition of machine consciousness, namely conscious machines must be able to see the future. This aspect is not totally unknown because it is used in creating the goal- and utility-based agents [Russell and Norvig, 1995, pp. 42-45], but we will go further than that.

Our approach for machine (self-) consciousness supposedly will be very compute intensive, so it may be interesting that in [Bátfai, 2009] we outlined an idea about where the necessary computations should be performed in the case of the operating systems.

2 Machine Consciousness

First, we give the general frames of definitions in which the term "computer program" is interpreted broadly, that is the Turing-like machines, the various web applications, the command-line interfaces, the GUIs, the kernel of operating systems and goal- or utility-based agents are regarded as computer programs.

Definition 1 (Knowing the Future Input). *A computer program knows its future input if it can predict that better than a random guess.*

Definition 2 (Knowing the Future State). *A computer program knows its future state if it can predict that better than a random guess.*

Definition 3 (Conscious Computer Programs). *The behavior of a computer program is considered conscious if it knows its future input.*

Definition 4 (Self-Conscious Computer Programs). *The behavior of a computer program is considered self-conscious if it knows its future state.*

Definition 5 (Intuitive Computer Programs). *The behavior of a conscious computer program is considered intuitive if its operation is based on its own predicted input rather than the real input.*

It is obvious that the consciousness is not an a priori property by our discussion. In addition, several levels of consciousness should be examined in given time intervals. Typically, the examination has two aspects, first we must study the source code. Second, we must observe the operation of the program. These remarks also indicate that our definitions are framed at a very, very high abstract level, in the concrete cases we probably will need to apply some inner simulation like the one introduced in [Hesslow and Jirenhed, 2007]. In conclusion, as regards the fulfillment of the definitions set out above, developers will obviously need to use sophisticated functional structures in the particular cases.

2.1 Some Intuitive Examples for Definitions

The intuitive usage of the definitions will be illustrated in this section. First, let's have a look at the following trivial examples in relation to the question of what data may be the input of a computer program. The input of a Turing machine is a word placed on its tape. The input of a CLI may be a set of commands entered by the user. The input of a GUI may be the set of user's activities. The input of a RoboCup [Kitano et al., 1997] agent is a set of information received from its aural (what it can hear), visual (what it can see) and body (what is its physical status) sensors. And finally, the activities of processes may be regarded as the input of the scheduler of an operating system.

But in the case of a Turing machine, interpreting of the term "knowing the future input" is worthy of further discussions, because the interpretation of its operation is not wholly straightforward. As an initial approximation, the concatenation of the former input words and the Turing machine in question should be given as an input to a "conscious" and modified universal Turing machine. Another approach is to apply a prefix Turing machine, where the current future input should be to the right of the input head on the unidirectional input tape.

Example 1 (Walking Across the Zebra Crossing). *This is a trivial example of daily life. Every day the author goes across the zebra crossing shown in Figure 1. Here I am standing (at a safe) 3-4 meters away from the kerb. Then I am going to start to go when the traffic light for cars has changed to yellow, because I know from former personal experience that the traffic light for pedestrians changes to green soon afterwards.*

This "conscious" behavior represents an advantage for the author over the other pedestrians, because while he are already on the move, others will be waiting for the green signal of the traffic light for pedestrians.

Figure 1: The author walks across the zebra crossing.

Example 2 (Conscious Stock Market Charts). *In a given time period, a stock market chart may be regarded as a conscious program if it can also successfully predict the time series of stock data in question. However, we can mention that another different type of input may be interpreted as further input data in an interactive case. For example, an interactive chart program may also examine and successfully predict the habits of its user. In general,*

we may remark that investigating the user's habits and patterns is an existing research field, for example see [Tseng et al., 2007], [Velayathan and Yamada, 2006] and [Blouin et al., 2011].

Example 3 (Conscious Databases). *In a naive approach, for example, the next couple of rows might be predicted by the table of a relational database that will contain the rows in question. But perhaps it makes more sense to think in terms of conscious behavior of OLAP systems, because the prediction is crucial to achieve good results in this field.*

Example 4 (Conscious Text Editors). *A successfully predicted input of a conscious text editor may be the next couple of words that will be entered by the user. In the world of text editing on mobile devices, it was a natural phase of development, for example, typing in predictive (iTap) mode is widely used in mobile phones to sending SMS or searching on the Web. We may notice that, nowadays, the investigation of several word prediction systems is an active research area, particularly with regard to mobile telephony [Kamvar and Baluja, 2007], [Agarwal and Arora, 2007].*

Example 5 (Self-Conscious Chatter Bots). *In this example, we are in a fortunate position that allows us to ask directly the questions about (knowing) the future. In compliance with this, we asked chatterbot Alice [Wallace, 2011], [Wallace, 2003], who won the Loebner Prize [Powers, 1998] 3 times in 2000, 2001 and 2004, the next questions about her future.*

```
Human: Hello, Alice!
ALICE: Hi there!
Human: I am writing a paper about machine consciousness,
          do you mind if I ask you a few questions?
ALICE: Will you include me in your essay?
Human: Yes, I will.
ALICE: I understand. Go ahead.
Human: What are you going to do tomorrow?
ALICE: You can find out if you become a botmaster.
Human: Will you be conversing with me tomorrow?
ALICE: I'll burn that bridge when I come to it. OK I will
          try to be conversing with you tomorrow.
```

If a conscious Alice had responded the following answer based on some analytics, our satisfaction would increase.

```
Human: What are you going to do tomorrow?
CX ALICE: I will be expected to talk to someone.
```

Example 6 (Self-Conscious RoboCup Agents). *A player agent may be regarded as a self-conscious program if it can run inner and simplified future*

simulation cycles of Soccer Server [Chen et al., 2002] and based on it can also successfully predict the future state of itself.

Example 7 (Intuitive RoboCup Agents). *In the situation shown in Figure 2, a self-conscious RoboCup agent may be regarded as an intuitive program if it can pass the ball to an open space so that a teammate able to reach it as an unexpected through pass. These are the magical moments of real football. Figure 2 schematically shows such a situation that happened in the match between FC Barcelona and Levante UD at the Camp Nou stadium on February 24, 2008 [YouTube, 2011], where the goal of Samuel Eto'o was assisted by Lionel Messi in the 55th min.*

2.2 The Trick of Consciousness

A computer program can be trivially made conscious, if we shift its virtual present to the true past. In other words this means that its all input has been delayed in time and in the meantime, we open a loophole to access input data of the present. It is a use case of the well-known conception of *"living in the past"*. This latter is described, for example, in [Hameroff, 2006], [Hameroff, 2003].

The following AspectJ Java code illustrates exactly this conception of time shifting. It is a simple game in which the two players P and Q try to catch the ball that moves with random walk on a field of fixed 80x24 size. Players win a point when they catch the ball.

```java
public class Game {

  public static final int FIELD_X = 80;
  public static final int FIELD_Y = 24;
  public static final int BALL_LIFESPAN = 1000;

  public static void main(String[] args) {

    final Ball ball = new Ball();
    final Player playerP = new Player(0),
    playerQ = new Player(FIELD_X - 1);

    int pointsP = 0, pointsQ = 0;
    for (int i = 0; i < BALL_LIFESPAN; ++i) {

      ball.move();

      new Thread() {
        public void run() {
```

Figure 2: This drawing is an illustration based on the match between FC Barcelona and Levante UD in the Primera División on February 24, 2008. It is created with the FerSML football simulation platform Bátfai [2010a], Bátfai [2010b].

```
        playerP.perception(ball.y);
    }
}.start();

new Thread() {
    public void run() {
        playerQ.perception(ball.y);
    }
```

```
    }.start();

    if (ball.x == 0 && playerP.y == ball.y) {
      ++pointsP;
    }
    if (ball.x == FIELD_X - 1 && playerQ.y == ball.y) {
      ++pointsQ;
    }

  }

  System.out.println(pointsP + " " + pointsQ);
  }
}
```

Listing 1: The source code for the Game class.

The ball can move all four directions with the same probability or, to be more precise, its movement is a random walk.

```
class Ball {

  int x = Game.FIELD_X / 2, y = Game.FIELD_Y / 2;

  void move() {
    int dx = (int) (Math.random() * 3) - 1;
    int dy = (int) (Math.random() * 3) - 1;

    if (x + dx < Game.FIELD_X && x + dx >= 0) {
      x += dx;
    }
    if (y + dy < Game.FIELD_Y && y + dy >= 0) {
      y += dy;
    }
  }
}
```

Listing 2: The source code for the Ball class.

The players can only move up and down on the sides of the field. They endeavor to catch the ball when it reaches the sides of the field. Our examples, the players P and Q are aware of the reality via an interface called Sensory.

```
interface Sensory {
  void perception(int ballY);
```

```
}
```

Listing 3: The source code for the Sensory interface.

```
class Player implements Sensory {

  int x = 0, y = Game.FIELD_Y / 2;

  public Player(int x) {
    this.x = x;
  }

  public void perception(int ballY) {
    move(ballY);
  }

  protected void move(int ballY) {
    if (y < ballY) {
      ++y;
    } else if (y > ballY) {
      --y;
    }
  }
}
```

Listing 4: The source code for the Player class.

The execution of the perception method of the interface Sensory is blocked for 500 millisecond by the following AspectJ aspect.

```
aspect Delay {

  public pointcut perceptionCall():
    call(public void Player.perception(int));

  before(): perceptionCall() {

    try {
      Thread.sleep(500);
    } catch(InterruptedException e){e.printStackTrace();}
  }
}
```

Listing 5: The source code for the Delay aspect.

It may be noted, as a curiosity, that using the 500 millisecond duration in the inserted code snippet was suggested by [Penrose, 1989] which presents

Libet and Kornhuber's results on the timing of consciousness [Libet et al., 1979], [Kornhuber et al., 1976]. But it is immaterial in this case where the results were observed are shown in Table 1. In addition, our *living in the past* aspect implementation is sufficiently buggy, for example, it has no mutual exclusion for protecting scores and coords. Nevertheless, this simple example delivers the expected results, namely that the scores decrease as we increase the amount of delay time.

Table 1: Execution results of the delay aspect (with the variable BALL-_LIFESPAN set to 100.000).

Naive example of the *living in the past*		
Time [ms]	**Aver. Scores**	**Exec. Time [min]**
javac	**1088**	9.7
javac	**1156**	10.0
no aspect	**1162**	9.6
no blocking	**1036**	9.7
0.001	**249**	11.2
0.01	**246**	10.0
0.1	**239**	10.2
0.5	**227**	9.9
1	**226**	10.0
2	**183**	9.9
5	**142**	10.1
50	**68**	11.3
200	**57**	16.9
500	**53**	23.5
1000	**52.5**	33.7

The reader can easily see that our example aspect does not contain any loopholes and any analytic codes, either. But, for example, in the case in which the movement of the ball is smooth (that is well predictable) writing some successful analytics and prediction codes, of course, could be trivial.

2.2.1 An Evolutionary Aspect of the *living in the past*

Why may this approach be interesting from the point of view of the awakening of consciousness? Because it may be possible that living matter could

have developed such *living in the past* aspects, in which they can run analytics and prediction methods. Doing so can start the evolutionary process simply and solely because the organisms who make wrong predictions become extinct. In this sense, applying *living in the past* offers an ability for the organisms to develop a successful prediction mechanism, since a predicted, interesting event that occurs in the "delay window" shown in Figure 3 can be effectively verifiable, because it has already happened.
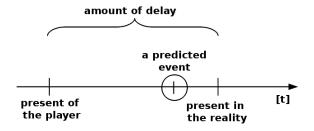

Figure 3: A simple schematic drawing of the well-known conception of *"living in the past"*.

Finally, it may be mentioned that the most recent sensational and paradoxical results of precognition [Bem, 2010] perhaps may be easily explained in the context of the *living in the past.*

2.3   The Consciousness as a Computing Paradigm

In the world of computer programs, the barbarian methods of natural selection may be partially waived because computers have the necessary computing resources that they can subsequently execute analytical computations, that is, for developing good solutions it is not necessary to extinct whole generations or races of living species.

In our opinion, a paradigm shift is needed to achieve the age of intelligent machines. The base of a new paradigm may be using our simple definitions of machine consciousness, that may be called consciousness oriented programming.

3   Consciousness Oriented Programming

Consciousness oriented programming is a new way of approaching software development, in which two basic situations can be distinguished today.

- Existing computer programs should be further developed to be conscious or self-conscious computer programs in line with our previous

definitions. In general case, it is nearly impossible to modify computer programs, but the situation is not hopeless if modifications are implemented as new aspects in the sense of AOP [Kiczales et al., 1997].

- New computer programs should be developed in conformity with the spirit of our definitions.

In both cases, the development of consciousness will require using prediction methods and the development of self-consciousness will require applying inner simulation in the sense mentioned in Section 2.

4 Use Cases for the COP

In this section, we follow the spirit of the definitions outlined previously.

4.1 Programming on Paper

Notation 1 (Predicted and Real Input). *Denote $\langle p(redicted)_i \rangle : N(\subseteq \mathbb{N}) \to I(nput)$ the sequence of the predicted input and $\langle r(eal)_i \rangle : N(\subseteq \mathbb{N}) \to I(nput)$ the sequence of the real input, where I denotes the set of all possible inputs.*

Definition 6 (Consciousness Indicator Sequence). *We define the consciousness indicator sequence $\langle c_i \rangle : N(\subseteq \mathbb{N}) \to \{0, 1\}$ as follows*

$$c_i = \begin{cases} 0 & \text{if } p_i \neq r_i, \\ 1 & \text{if } p_i = r_i. \end{cases}$$

Definition 7 (Conscious Computer Programs). *In a given time interval, the behavior of a computer program is referred to as conscious if its consciousness indicator sequence is not Kolmogorov-Chaitin random [Li and Vitányi, 2008].*

We should remark that this definition does not kill the consciousness, because the Kolmogorov-Chaitin randomness is algorithmically undecidable.

The next section will diverge from the proposed inner prediction mechanism to a simpler way, and meanwhile we will stay within the classical framework of Turing machines.

4.1.1 Quasi-Intuitive Machines and Languages

In the majority of cases in this subsection, a comma between words denotes the concatenation of these words which are suitable encoded if necessary.

Definition 8 (Universal Quasi-Intuitive Machines). *Let T be a Turing machine and let p be a positive real number. An universal quasi-intuitive machine $Q_{x,p}$ is created by the following scheme shown in Figure 4, provided that there exists a sequence of words $x_1, \ldots, x_n(= x)$ having the properties that*

$$i = 1, \qquad T(x_i) = yes \tag{1}$$

$$2 \le i \le n, \quad Q_{x_{i-1},p}(T, x_i) = yes \tag{2}$$

In Figure 4, U denotes an universal Turing machine and "$d(x,y) < p$" denotes a Turing machine that can indicate that the input words x and y are similar to each other. If this machine $Q_{x,p}$ stops it makes the computation of the function $Q_{x,p}(T, y) = QIM(x_1, \ldots, x_{n-1}, x, y, T, p)$, $QIM : \{0,1\}^{*n+3} \to \{yes, no\}$. The architectural model for the machine QIM is shown in Figure 5.
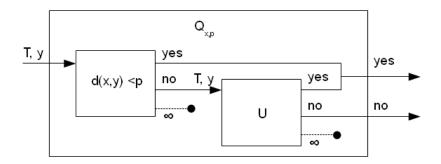

Figure 4: An architectural model for the universal quasi-intuitive machine.

Remark 1 (Comparisons of the Words). *Using the normalized information distance (NID) [Li et al., 2003] as the metric distance $d(x,y)$ of the two words x and y is theoretically exciting, but in this case we cannot build the Turing machine $Q_{x,p}$ that contains such a "$d(x,y) < p$" machine, because this is not an existing machine due to the Kolmogorov complexity function is not partial recursive [Li and Vitányi, 2008, pp. 127, pp. 216]. Specifically, the computability of NID is discussed in [Terwijn et al., 2011, 2010]. But then we can successfully use the normalized compression distance (NCD) [Li et al., 2003, Vitányi et al., 2008, Cilibrasi and Vitányi, 2005] instead of the theoretical normalized information distance because the compression distance is trivially partial recursive [Terwijn et al., 2011]. Another trivial option may be to use the Google similarity distance (NGD) [Cilibrasi and Vitányi, 2007] or the normalized web distance (NWD) [Cilibrasi and Vitányi, 2009] as the metric d. In the following, we suppose that the predicate $d(x,y) < p$ is recursive.*

Remark 2. *In the intuitive sense, $x \in \{0,1\}^*$ is such a word that has already been accepted by the Turing machines T or Q.*

Definition 9 (The Universal Quasi-Intuitive Language)**.** *The universal quasi-intuitive language*

$$QIL = \{x_1, \ldots, x_n(=x), y, T, p \mid T \text{ is defined}, T(x_1) = yes,$$
$$Q_{x_{i-1},p}(T, x_i) = yes, 2 \leq i \leq n \text{ and } Q_{x,p}(T, y) = yes\}.$$

Theorem 1. $QIL \in \mathcal{RE}$.

Proof. To verify assertion $QIL \in \mathcal{RE}$, it is sufficient to observe that the language accepted by the machine QIM shown in Figure 5 is equal to QIL. (We believe that we can prove a bit stronger theorem $QIL \in \mathcal{RE} \setminus \mathcal{R}$.) □
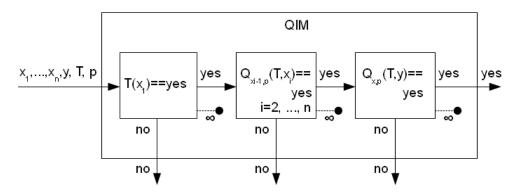

Figure 5: An architectural model for a machine QIM that accepts the language QIM.

Definition 10 (Similar Languages)**.** *Let $E \subseteq \{0,1\}^*$ be a given language and let $p \in \mathbb{R}$ be a positive real number. The language $SL_E = \{y \mid y \in \{0,1\}^*, d(x, y) < p, x \in E\}$ is said to be similar to E.*

Theorem 2. $E \in \mathcal{RE} \Rightarrow SL_E \in \mathcal{RE} \setminus \mathcal{R}$.

Proof. In the case $E \in \mathcal{R}$, we can construct a new Turing machine SLM shown in Figure 6 which accepts SL_E.

To see why the language SL_E is not recursive, consider the case of $y \notin SL_E$, it is possible that the new machine SLM will never halt, because it is possible that the part labelled "For all x/d(x, y)<p" will continue searching for suitable x for ever. It may be noted that the the canonical ordering of $\{0,1\}^*$, for example shown in [Rónyai et al., 2004], can be applied to help to enumerate the words of the language E by the part labelled "For all x".

In the case $E \in \mathcal{RE} \setminus \mathcal{R}$, we can construct a new Turing machine SLM' shown in Figure 7 which accepts SL_E. It may be also noted here that a
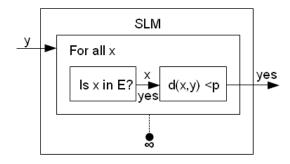
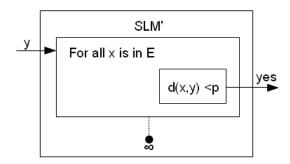Figure 6: The Turing machine SLM for the case $E \in \mathcal{R}$.

Figure 7: The Turing machine SLM' for the case $E \in \mathcal{RE} \setminus \mathcal{R}$.

procedure based on Cantor's first diagonal argument, for example shown in [Rónyai et al., 2004], can be applied to enumerate the words of the language E by the part labelled "For all x is in E". $\qquad \square$

Definition 11 (Quasi-Intuitive Languages)**.** *With the notation already introduced in Definition 8, a quasi-intuitive language $QILT$ is defined as $QILT \ (= \cup_{i=1}^{\infty} QILT_i) = \cup_{i=1}^{\infty} I_{p,i}^{T}$, where $I_{p,i}^{T}$ is constructed by the following scheme:*

$$I_{p,1}^{T} \ = \ \{x \,|\, T(x) = yes\} \tag{3}$$
$$I_{p,i+1}^{T} \ = \ \{y \,|\, x \in I_{p,i}^{T}, Q_{x,p}(T,y) = yes\}, i \geq 1 \tag{4}$$

where $y \in \{0,1\}^*$ is an arbitrary word or, for example, $y \in L(G)$ generated by some generative grammar G.

Theorem 3. *Let $n \in \mathbb{N}$ be a given natural number and let T be a Turing machine. Then $L_T \in \mathcal{RE} \Rightarrow QILT_n \in \mathcal{RE} \setminus \mathcal{R}$.*

Proof. Let us observe that

$$I_{p,1}^{T} \ = \ L_T \tag{5}$$
$$I_{p,i+1}^{T} \ = \ L_T \cup SL_{I_{p,i}^{T}} \tag{6}$$

16

$\square$

Definition 12 (Similar Words). *Let $L \subseteq \{0,1\}^*$ be a given language and let $p \in \mathbb{R}$ be a positive real number. We define the language of similar words as follows $SW_L = \{x, y \mid x \in L, y \in \{0,1\}^*, d(x,y) < p\}$.*

Theorem 4. $L \in \mathcal{R} \Rightarrow SW_L \in \mathcal{R}$.

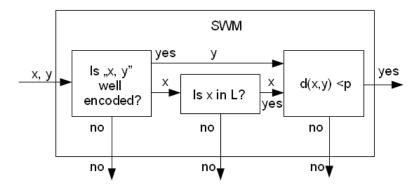*Proof.* We can construct a new Turing machine *SWM* shown in Figure 8 which accepts SW_L and always halts. $\square$

Figure 8: The Turing machine *SWM*.

Definition 13 (Quasi-Intuitive or Self-Similar Words). *Let T be a given Turing machine and let $s \in \{0,1\}^*$ be a word such that $T(s) = yes$. Finally, let $\langle r_i \rangle : N(\subseteq \mathbb{N}) \to \{0,1\}^*$ be a finite or infinite sequence of arbitrary words. The union of the elements of the sequence of sets I_i is said to be self-similar, if the sequence is created by the following method:*

$$
\begin{aligned}
I_1 &= \{s\} \\
i \geq 1, I_{i+1} &= \begin{cases} I_i \cup \{r_i\}, & \text{if } \exists x \in I_i : Q_{x,p}(T, r_i) = yes \\ I_i, & \text{otherwise.} \end{cases}
\end{aligned}
$$

Remark 3 (Consciousness Indicator Sequence). *In the case of the self-similar words, the consciousness indicator sequence may be interpreted as $c_n = \bigvee_{x \in I_{n+1}} Q_{x,p}(T, r_n)$.*

Example 8. *Let T be a Turing Machine, which accepts the language $\{a^n b^n c^n$*

17

$| n \geq 1\}$, $d = 1/4$ and the word $s = aaaaaabbbbbbcccccc = a^6 b^6 c^6$.

$$I_1 = \{s\}$$

$$
\begin{aligned}
r_1 &= aaaaaabbbbbbaaaaaa = a^6 b^6 a^6, I_2 = \{s\}, c_1 = no \\
r_2 &= aaaaaabbbbbbccc = a^6 b^6 c^3, I_3 = \{s, r2\}, c_2 = yes \\
r_3 &= aaaaabbbbbccccc = a^5 b^5 c^5, I_4 = \{s, r2, r3\}, c_3 = yes \\
r_4 &= a^7 b^7 c^7, I_5 = \{s, r2, r3, r4\}, c_4 = yes \\
r_5 &= c^1 a^5 b^5 c^4, I_6 = \{s, r2, r3, r4, r5\}, c_5 = yes \\
r_6 &= c^3 a^6 b^5 c^4 b^2, I_7 = \{s, r2, r3, r4, r5\}, c_6 = \mathbf{no} \\
r_7 &= c^3 a^6 b^5 c^3, I_8 = \{s, r2, r3, r4, r5, r7\}, c_7 = yes \\
r_8 &= r_6, I_9 = \{s, r2, r3, r4, r5, r7, r8\}, c_8 = \mathbf{yes}
\end{aligned}
$$

where the CompLearn package [Cilibrasi, 2011] is used to compute NCD:

$NCD(s, r_1) = 0.3125$

$NCD(s, r_2) = 0.176471$

$NCD(s, r_3) = 0.25, NCD(r_2, r_3) = 0.294118$

$NCD(s, r_4) = 0.4375, NCD(r_2, r_4) = 0.470588,$

$\qquad NCD(r_3, r_4) = 0.5625$

$NCD(s, r_5) = 0.25, NCD(r_2, r_5) = 0.235294,$

$\qquad NCD(r_3, r_5) = 0.1875, NCD(r_4, r_5) = 0.25$

$NCD(s, r_6) = 0.35, NCD(r_2, r_6) = 0.3, NCD(r_3, r_6) = 0.3,$

$\qquad NCD(r_4, r_6) = 0.3, NCD(r_5, r_6) = 0.25$

$NCD(s, r_7) = 0.263158, NCD(r_2, r_7) = 0.210526,$

$\qquad NCD(r_3, r_7) = 0.210526, NCD(r_4, r_7) = 0.210526$

$\qquad NCD(r_5, r_7) = 0.157895$

$NCD(r_7, r_8) = 0.15$

We should remark that the symmetry of NCD may be violated among short words [Cilibrasi and Vitányi, 2005]. For example, $NCD(a^7 b^7 c^7, a^6 b^6 c^3) = 0,235294 \neq 0,470588 = NCD(a^6 b^6 c^3, a^7 b^7 c^7)$.

4.2 Programming on Computer

In the world of real programming, we do have plan to develop such APIs which can be used to successfully implement our definitions of machine consciousness. For some given types of applications, we are going to investigate the development of a suitable, open source Java and AspectJ APIs to enable *seeing the future*.

4.2.1 ConsciousJ

Designing a new programming language is another exciting possibility. At the conceptual level, the following ConsciousJ code snippet illustrates the usage of two new keywords "conscious" and "predicted", though certainly the language ConsciousJ does not exist yet. In our case, this new language to be developed can be imagined as an extension of Java or AspectJ.

```
conscious class Player {
  int x = 0, y = Game.FIELD_Y / 2;

  protected void move(predicted int ballY) {
    if (y < ballY) {
      ++y;
    } else if (y > ballY) {
      --y;
    }
  }

}
```
Listing 6: A "conscious" code snippet written in a fictitious language called ConsciousJ.

In practice, this code snippet shows that the uncertainty is appeared at the level of the programming language. A ConsciousJ class consists of attributes and methods, plus it may contain predicted attributes. This conception is shown in the following fictitious code snippet in Listing 7.

```
conscious class Player implements Runnable {

  int x = 0, y = GameS.FIELD_Y / 2;
  int points = 0;
  Ball ball;

  int time = 0;
  int next = 6;
  int ballYHelper = 0;

  org.consciousj.pred.KalmanFilter kp;
  org.consciousj.pred.ARMA ap;
  org.consciousj.pred.SimplePast sp;

  predicted org.consciousj.primitivetype.Int ballY;

  public pointcut moveCall():
    call(protected void move(predicted int ballY));
```

```
input(): moveCall() {

  this.ballY.receive(ballY);

  if (++time < 1000) {
      this.ballY.method(sp);
  } else if (time < 5000) {
      this.ballY.method(ap);
  } else {
      this.ballY.method(kp);
  }

  if (time % next == 0) {
      ballYHelper = this.ballY.next();
  }
  ballY = ballYHelper;
}

public Player(int x, Ball ball) {
  this.x = x;
  this.ball = ball;

  new Thread(this).start();

}

public void run() {

  for (;;) {

    move(ball.y);

    try {
      Thread.sleep(20);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

    if (ball.y < 0) {
      break;
    }
  }
}
```

```
    protected void move(predicted int ballY) {

        if (y < ballY) {
            ++y;
        } else {
            --y;
        }
    }

}
```
Listing 7: A "conscious" class written in ConsciousJ.

5 Conclusion

The idealized objective of COP is to integrate the agent-based approach into daily software development practices. What is more, the agents should be able to see the future, or, using words of Alan Turing's [Turing, 1995] essay, everyday softwares should be able to *see a short distances ahead*. Naturally these words by Turing were really meant for humans.

The present paper presented an overall conceptual framework so that it could contribute to the attainment of this objective.

6 Acknowledgements

References

S. Agarwal and S. Arora. Context based word prediction for texting language. In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, RIAO '07, pages 360–368, Paris, France, France, 2007. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE. URL `http://portal.acm.org/citation.cfm?id=1931390.1931426`.

B. J. Baars and S. Franklin. Consciousness is computational: the lida model of global workspace theory. *International Journal of Machine Consciousness*, 01(01):23, 2009. URL `http://www.worldscinet.com/ijmc/01/0101/S1793843009000050.html`.

N. Bátfai. A conceivable origin of machine consciousness in the idle process. *CoRR*, abs/0909.5064, 2009. URL `http://arxiv.org/abs/0909.5064`.

N. Bátfai. Footballer and football simulation markup language and related simulation software development. *Journal of Computer Science and Control Systems*, 3(1):13–18, 2010a. URL `http://electroinf.uoradea.ro/reviste\20CSCS/documente/JCSCS_2010/JCSCS_Nr_1_integral.pdf`.

N. Bátfai. The socceral force. *CoRR*, abs/1004.2003, 2010b. URL `http://arxiv.org/abs/1004.2003`.

D. J. Bem. Feeling the future: Experimental evidence for anomalous retroactive influences on cognition and affect. *Journal of Personality and Social Psychology*, 100(3):407– 425, 2010. URL `http://doi.apa.org/getdoi.cfm?doi=10.1037/a0021524`.

A. Blouin, B. Morin, O. Beaudoux, G. Nain, P. Albers, and J.-M. Jézéquel. Combining aspect-oriented modeling with property-based reasoning to improve user interface adaptation. In *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '11, pages 85–94, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0670-6. doi: http://doi.acm.org/10.1145/1996461.1996500. URL `http://doi.acm.org/10.1145/1996461.1996500`.

M. Chen, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummenje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, and X. Yin. *RoboCup Soccer Server*. The RoboCup Federation, 2002. URL `http://sserver.sourceforge.net`.

R. Cilibrasi. Complearn, 2011. URL `http://www.complearn.org/`.

R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

R. Cilibrasi and P. M. B. Vitányi. Normalized web distance and word similarity. *CoRR*, abs/0905.4039, 2009.

R. L. Cilibrasi and P. M. B. Vitányi. The google similarity distance. *IEEE Trans. on Knowl. and Data Eng.*, 19:370–383, March 2007. ISSN 1041-4347. doi: 10.1109/TKDE.2007.48. URL `http://portal.acm.org/citation.cfm?id=1263132.1263333`.

S. Hameroff. Time, Consciousness and Quantum Events in Fundamental Spacetime Geometry. In *The nature of time: Physics, geometry and perception*. Proceedings of a NATO Advanced Research Workshop, 2003.

S. Hameroff. Consciousness, Neurobiology and Quantum Mechanics: The Case for a Connection. pages 193–253. 2006. doi: 10.1007/3-540-36723-3_6. URL `http://dx.doi.org/10.1007/3-540-36723-3_6`.

22

G. Hesslow and D.-A. Jirenhed. Must machines be zombies? internal simulation as a mechanism for machine consciousness. In *AI and Consciousness: Theoretical Foundations and Current Approaches*, Proceedings of the AAAI fall symposium on machine consciousness, page 78?83. AAAI Press, 2007. URL `http://www.aaai.org/Papers/Symposia/Fall/2007/FS-07-01/FS07-01-014.pdf`.

M. Kamvar and S. Baluja. The role of context in query input: using contextual signals to complete queries on mobile devices. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, pages 405–412, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-862-6. doi: http://doi.acm.org/10.1145/1377999.1378046. URL `http://doi.acm.org/10.1145/1377999.1378046`.

G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. marc Loingtier, and J. Irwin. Aspect-oriented programming. In *ECOOP*. SpringerVerlag, 1997.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, AGENTS '97, pages 340–347, New York, NY, USA, 1997. ACM. ISBN 0-89791-877-0. doi: http://doi.acm.org/10.1145/267658.267738. URL `http://doi.acm.org/10.1145/267658.267738`.

H. H. Kornhuber, L. Deecke, and P. Scheid. Voluntary finger movement in man: Cerebral potentials and theory. *Biological Cybernetics*, (23), 1976.

M. Li and P. M. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008. ISBN 0387339981, 9780387339986.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 863–872, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. ISBN 0-89871-538-5. URL `http://portal.acm.org/citation.cfm?id=644108.644253`.

B. Libet, E. Wright, B. Feinstein, , and D. K. Pearl. Subjective referral of the timing for a conscious sensory experience. *Brain*, (102):193–224, 1979.

R. Penrose. *The emperor's new mind: concerning computers, minds, and the laws of physics*. Oxford University Press, Inc., New York, NY, USA, 1989. ISBN 0-19-851973-7.

R. Penrose and S. Hameroff. Quantum computation in brain microtubules? the penrose-hameroff Orch OR model of consciousness. *Philosophical Transactions Royal Society London (A)*, 356:1869–1896, 1998.

D. M. W. Powers. The total turing test and the loebner prize. In *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, NeMLaP3/CoNLL '98, pages 279–280, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. ISBN 0-7258-0634-6. URL `http://portal.acm.org/citation.cfm?id=1603899.1603947`.

L. Rónyai, G. Iványos, and R. Szabó. *Algoritmusok*. Typotex, Budapest, 2004.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 1995.

A. Sloman and R. Chrisley. Virtual machines and consciousness. *Journal of Consciousness Studies*, 10:4–5, 2002.

J. Starzyk and D. Prasad. A computational model of machine consciousness. *International Journal of Machine Consciousness*, in press, 2011. URL `http://www.ohio.edu/people/starzykj/network/Research/Papers/AComputationalModelofMachineConsciousness.pdf`.

S. Terwijn, L. Torenvliet, and P. M. B. Vitányi. Normalized information distance is not semicomputable. *CoRR*, abs/1006.3275, 2010.

S. A. Terwijn, L. Torenvliet, and P. M. B. Vitányi. Nonapproximability of the normalized information distance. *J. Comput. Syst. Sci.*, 77:738–742, July 2011. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/j.jcss.2010.06.018. URL `http://dx.doi.org/10.1016/j.jcss.2010.06.018`.

V. S. Tseng, K. W. Lin, and J.-C. Chang. Prediction of user navigation patterns by mining the temporal web usage evolution. *Soft Comput.*, 12:157–163, September 2007. ISSN 1432-7643. doi: 10.1007/s00500-007-0190-y. URL `http://portal.acm.org/citation.cfm?id=1290506.1290511`.

A. M. Turing. *Computing machinery and intelligence*, pages 11–35. MIT Press, Cambridge, MA, USA, 1995. ISBN 0-262-56092-5. URL `http://portal.acm.org/citation.cfm?id=216408.216410`.

G. Velayathan and S. Yamada. Behavior-based web page evaluation. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 841–842, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: http://doi.acm.org/10.1145/1135777.1135905. URL `http://doi.acm.org/10.1145/1135777.1135905`.

P. M. B. Vitányi, F. J. Balbach, R. Cilibrasi, and M. Li. Normalized information distance. *CoRR*, abs/0809.2553, 2008.

R. Wallace. *The Elements of AIML Style*, 2003. URL `http://www.alicebot.org/style.pdf`.

R. Wallace. Alicebot, 2011. URL `http://alicebot.blogspot.com`.

E. P. Wigner. Remarks on the mind-body question. *Symmetries and Reflections*, page 179, 1967.

YouTube. The match between FC Barcelona and Levante UD at the Camp Nou stadium on february 24, 2008, 2011. URL `http://www.youtube.com/watch?v=oTIEpqMwCbk,\from\3:20\to\3:30,http://www.youtube.com/watch?v=VfJT2U3OUps,\from\1:00\to\1:04,\http://www.youtube.com/watch?v=wkUYcOPBoz8,\from\0:19`.