The algorithm of the Thinking machine

Dimiter Dobrev

Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
d@dobrev.com

Abstract. In this article we consider the questions 'What is AI?' and 'How to write a program that satisfies the definition of AI?' It deals with the basic concepts and modules that must be at the heart of this program. The most interesting concept that is discussed here is the concept of abstract signals. Each of these signals is related to the result of a particular experiment. The abstract signal is a function that at any time point returns the probability the corresponding experiment to return true.

Introduction

Throughout my life I've tried to answer the question 'What is AI?' and write the program that is AI. I've already known the answer to the question 'What is AI?' for 16 years now but the AI algorithm has eluded me. I've come up with individual fragments but something has always been missing to put the puzzle together. Finally, I gathered all the missing pieces and I can introduce you to this algorithm. That is, in this article you will find a sufficiently detailed description of the algorithm of the AI. That sounds so audacious that probably you won't believe me. Frankly, even I do not fully believe myself. I'll believe it only when someone writes a program executing this algorithm and when I see that this program actually works. Even if you don't manage to believe in the importance of this article, I hope that you will like it.

We will first deal with the question 'What is AI?' This question has already been answered to by Alan Turing 66 years ago [1]. The definition of AI has a small addition made by me in the year of 2000 [2]. Then we will describe the problem in formal terms, because if you want to write a specific program, it must be clear what its input and output is supposed to be. We will also give a concrete example, which although simple is very significant for understanding of the problem. After this preparation, we will begin with the description of the concrete algorithm and its modules. To understand the world we should understand the dependencies in this world. The simplest dependencies are the dependencies without memory. These are the dependencies like: 'If I see this and do that that something will happen.' That is, dependencies without memory are the link between the last step and the next one. We will generalize and consider that dependencies without memory are the link between the last couple of steps and the next one. These dependencies will be presented as a conjunction

of literals and we will find them with brute force by doing statistics of a huge number of conjunctions. The next issue we will consider will be the finite-memory dependencies. They will be represented as finite-state automata. We cannot find these automata by brute force and therefore will have to use some tricks to find them.

Should we seek for dependencies with infinite memory? Here is an example of such dependence: 'If until now A has happened more times than B, then something.' People can not detect such dependencies and therefore we will assume that AI can also do without them.

Further in the article we will deal with the question of abstract signals. This is the last piece of the puzzle that was missing to solve the problem. These signals objectively exist in the world, though we can not directly observe them. Their value may indirectly be established by an experiment. The important thing for these signals is that they give us a full description of the state of the world.

The next concept that we consider is the concept of the multi-agent world. It turns out that the only way to understand a complex world is to set-off individual parts of it in black boxes that we will call agents. Of course, we will not deal with how these agents are designed internally and will only predict their behavior. For example, we can assume that a given agent is our enemy and expect it to always do the worst. (Look at [6])

The last module, which we'll discuss, is the module of centralized planning. The plan this module will create is a sequence of goals. The machine will head to the first goal, which will be much easier than directly heading to the final goal. (Look at [7])

What distinguishes this article from most articles on AI? We see AI as a transducer, while most of our colleagues see it as a function. The differences are many. First, the function has no memory and second – it recognize any input information by one step. Essentially, most articles in the field of AI work on the problem of how to find a detector function form a set of input examples. Basically, this is the problem of an approximation of a function. An example of such articles is the works in the field of neural networks. It is funny how such a simple task has produces so poor results. The main problem is that recognition is not usually a one-step process. Here's an example: You see someone who looks like your dear old friend Peter, but you are not sure. You wave your hand and he waves back. So it is Peter. That is, recognition is a multi-step process involving a number of experiments and therefore we must look for a transducer, rather than a function.

What is AI?

The first definition of AI is given by Alan Turing and it says: 'If you put behind curtain a human and a machine, and you talk to them and you can not reliably tell the machine from the human, then this machine is AI.'

There are many criticisms directed at this definition. Some say it is informal, while others say it is subjective because it depends on the judgment of the

evaluator. This definition is really informal, but this is not a problem. There is also a certain element of subjectivity, but that is not a problem also. When we are examining students there is also an element of subjectivity. However, we say that a good student is the one who has taken the exams and we do not worry about the subjectivity of the definition of a good student.

Another criticism of the definition of Turing is that it raises the bar too high. Some people believe that we must want much less from AI. The problem with these people is that they do not believe in the existence of AI and therefore they want the definition of AI to define not something that does not exist, but something else, something simpler – something that has already been invented.

All these criticisms are completely unfounded. In the definition of Turing there is only one problem and that is that it defines something more than AI.

If I tell you 'Imagine a beer bottle', you will imagine a bottle full of beer. This is your definition, but it would more proper to imagine an empty bottle because the full bottle is something more than what we want to define.

The same problem lies with the definition of Turing. It defines AI plus education. AI without education is like an empty beer bottle, but just as we know how to fill an empty bottle of beer, so we know how to educate the uneducated AI.

We need a definition of artificial intelligence that defines AI without education. I proposed such a definition in 2000 [2]. It reads: 'AI is this program that in any world would do no worse than a man.'

This definition does not depend on education because the machine just like a human can be born in a different world and get educated differently, so the program can be run in a different world. The launch of the program corresponds to the birth in humans.

To formalize this definition we should say what a world is, when one is doing better than another, and the most difficult of all is to say how high we will raise the bar, which means to say how smart the program must be. If we say 'not more stupid than a human', this would be quite informal. At least, there are smart and dumb people to a different extent, although the difference in intelligence between the smartest and the most stupid person is not that great.

In order to avoid comparison with humans in [3] and [4], we've introduced the concept of IQ, which is a number which can be calculated for each program. AI are those programs whose IQ is above a certain value. We cannot say exactly how high this value has to be, as we cannot say what the minimum score for admission to the University next year will be.

This article will not consider the question 'How intelligent must AI be?' but will describe a program that is intelligent enough.

Formulation of the problem

We shall assume that we have a discrete time and that the points in time are natural numbers. At each step the machine will receive input (information) from the world and will produce an outcome (action) back to the world.

We shall assume that each step consists of two points in time because later we will look at some finite automata that will snap twice a step (once when the machine gets an input and once when an output is generated). They snap, i.e. change their state.

We shall assume that the input is described by two functions ($View$ and $Score$), and the output – by one function ($Action$). We shall assume that the input enters at the even points in time and that the output is generated at odd points in time. That is, the first two functions shall be defined only for the even numbers, and the $Action$ function will be defined only for odd ones. We've split the input into two functions because we shall assume that the value of $Score$ has a meaning that does not depend on the particular world but is common for all worlds. On the contrary, the meaning of the $View$ function depends entirely on the specific world, and our machine will have to navigate in this world and understand what the meaning of the information it receives from the $View$ function is.

What is the meaning of the $Score$ function? The possible values of this function will be the constant $Nothing$ or a number. The aim of our machine will be to sum up the largest possible average of all the results of the $Score$ function, which are different from the constant $Nothing$.

We shall assume that the function $View$ returns an n-tuple of scalars, the $Action$ function, respectively m-tuple of scalars, and the $Score$ returns a single scalar ($Nothing$ or a number).

In [2] we assumed that the input and output are Boolean vectors because in this way we can encode any final information. Later we decided it was better to avoid unnecessary coding because the goal is to understand the world, and an additional coding can make the world more difficult to understand. Now we will assume that the scalar functions are final. We could generalize and assume that the value of the scalars can be even a whole or real number, but this article will not do this generalization and we shall limit ourselves to finite scalars.

The arbitrary world comprises a set of internal states S, (one of which, s_0, is initial) and the functions $World$, $View$ and $Score$. The $World$ function tells us what will be the next state of the world on the next step.

$s_{i+2} = World(s_i, a_{i+1})$

Here a_{i+1} is the vector that returns from $Action(i+1)$. The $Action$ function is not part of the world, its values are the actions of our machine. The $World$ function is defined only for even values of i because one step is made of two points in time.

The $Score$ and $View$ functions take s_i for a value and return an n-tuple for the first one and a one scalar for the second one.

We will assume that the function $World$ is not determined and that given a certain argument it may return different states, each of which has a non-zero probability to be returned. In the private case, when $World$ is determined for a given argument, it will return exactly one specific state and will return it with a probability of one.

Incorrect Moves

We will further assume that the $World$ function is not total. That is, not every move is possible. In particular state some moves will be correct, others will not be correct. It is better to assume that each state has at least one correct move. That is, to assume that there is no lock-up where the machine cannot make any further moves. If such lock-ups exist, we could easily associate them with the end of life (no further moves possible). When life happens to be shorter, we will assess it to the point it has reached. The natural heuristics of the machine will be to avoid lock-ups and generally seek situations where there are more possible moves. This instinct is natural to humans as well. They do not like narrow, confined spaces; do not like to be tied down, etc.

We will permit the machine to try incorrect moves and will not punish it for it. Normally, incorrect moves are used to collect information. For example, people grope in the dark and touch the walls to find their way. The human hand can not pass through the wall, and that's why we could say that groping the wall is an incorrect move.

Incorrect moves will not change the state of the world and will not increase the counter of steps t. The machine has nothing to lose when it tries incorrect move. It will also not gain anything except acquiring the information that this move has been an incorrect one.

Specific example

To get out of the swamp of abstraction we will take a specific example. Let's look at the world where the machine plays the game Tic-Tac-Toe.

The life of the machine should be long enough for it to have time to learn how to play. So we will assume that it does not play just one game but that after each game the panel is cleared and a new game is started.

The machine will not see the entire board of the game; it will see only one of the cells. That means that it will have one eye that can move along the board and the machine will only see the cell on which the eye is positioned.

The operation of the machine will consist of a 3-tuple consisting of:
$< horizontal\_move, vertical\_move, marking\_by\_X >$
Each of these three coordinates can take one of the following values:
$horizontal\_move \in \{to\_the\_left, to\_the\_right, stay\_where\_you\_are\}$
$vertical\_move \in \{upwards, downwards, stay\_where\_you\_are\}$
$marking\_by\_X \in \{mark\_by\_X, do\_not\_mark\}$

That means that the eye will be able to move in all directions, even diagonally (but only by one step). The machine will be able to mark by X but only the cell on which the eye is positioned on.

We must point out that not all moves are correct. For example, when we are in the left column, the move to the left will be incorrect (regardless of the other two coordinates of the output). The move 'mark by X' will also be incorrect except in cases where the eye is positioned on an empty cell. The incorrect moves

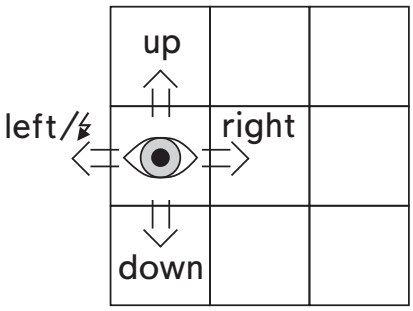

Fig. 1. Specific example.

will carry valuable information. For example, we will know that we are in the left column by the fact that the move to the left is incorrect.

The $View$ and $Score$ functions will return scalars

$View(s_i) \in \{X, O, empty\}$

$Score(s_i) \in \{win, loss, draw, Nothing\}$

Since we said that the $Score$ values must we numbers, we will replace win, loss and draw by the numbers 1, -1, 0 but will keep in mind the meaning of these numbers.

The internal state si of this world consists of a 3x3 board with nine cells, each of which can take three possible values plus the coordinates of the eye, plus the $Score$ results for this state. We have to add the latter because when the game is over the board is cleared and is empty, but the value of $Score$ is determined by the previous state on the board, the last move of the machine and possibly the move of the imaginary opponent.

What happens when the machine marks an X? The imaginary opponent responds by marking an O in one of the empty cells. We shall assume that this happens immediately, on the same step, and that the very next step we can see the O marked by the imaginary opponent (as long as we moved the eye to that cell). In cases where the game ends after the move of the machine or after the move of the imaginary opponent, the board is cleared and $Score$ returns 1, -1 or 0, depending on how the game ended.

This seems to be enough to describe the world in which the machine plays Tic-Tac-Toe. This world seems absolutely simple and there is a big chance the machine understands it completely. Still, we miss something very important and it's the imaginary opponent. He is a part of the world because in the world there is someone who plays against the machine. That someone is the most complex and the most difficult concept of the world.

Imagine that the imaginary opponent is a professor of mathematics or a student in love. It is assumed that the actions of the professor are quite logical and it would not be difficult to understand and predict his next move. Much more difficult will be with the student in love because she will play illogically

and unpredictably. Below we will see that we can take the imaginary opponent out of the model of the world and thus the model will greatly simplify.

The conclusion is: Although the world of Tic-Tac-Toe game is quite simple, it has all the elements that are important to the AI program. If our program manages to cope with this world, there is a chance it could cope in any world, which would mean that it actually meets the definition of AI. Of course, we should not palter. Our program should make out the rules of the game all by itself and learn how to play without us embedding the rules in some way in the program in advance.

History

We shall refer to 'History' as the values of the functions $View$, $Score$ and $Action$ in the range of 0 to t (from birth to the current moment). Along with these three functions, we will also add the function $IncorrectMoves$, which for each even t will return the set of incorrect moves that the machine has unsuccessfully tried at the moment t. $IncorrectMoves$ will return a set and not a list because it does not matter in what order these moves have been made.

On the basis of the history, the machine will collect statistics and will try to understand the world. Based on the last few moves from the history, it will determine the true values of the statements without memory. On the basis of the entire history, it will determine the true values of finite-memory statements. We will not consider infinite-memory statements.

Statements without memory are conjunctions from past literals. Past literals are literals whose value we already know, while future literals are those whose value is yet to be found out.

Definition: Each literal will be of the type: $View_i(t+k) = constant$, where $View_i$ is the i-th coordinate of the vector $View$, t is the current point in time, and $constant$ is a constant. Here, in place of the function $View$ can be taken by any of the functions $Action$ and $Score$.

Definition: Depending on the value of k, we will call the literals past or future. When $k \leq 0$, the literals will be past. If $k \geq 1$, the literals will be future and they will apply to the future.

Why will we discuss conjunctions only and not discuss the more complex Boolean functions? That's because every Boolean function can be represented as a disjunction of conjunctions of literals and therefore, if we examine the behavior of conjunctions, this will give us the behavior of all Boolean functions.

Basically, we are only interested in the future, but the only way to understand and predict the future is to analyze the past.

We will not assume that the machine remembers its entire history because it would need a lot of memory and also it would be quite laborious to analyze the whole history to extract some information.

What will be remembered from the history is a little buffer from the last k steps and a large amount of statistics for a huge number of statements without

memory. By huge number we mean millions and billions. I did a simple experiment [5], where the length of the conjunction was limited to four or five, yet the number of axioms rose to several million. I say axioms, not conjunctions, because I was then only interested in conjunctions that have always been a lie (i.e. their negations are axioms, at least so far). Now controversial conjunctions are still the most interesting, but now we will be interested in the others as well.

Another thing that will have to be remembered from the entire history is the current state of the finite-memory statements. Each of these statements is determined by the finite-state automaton. The current state of the automaton should be monitored and known (remembered).

Definition: The term 'Signal' will refer to any function whose argument is a point in time and whose value is a scalar.

We will further expand the set of literals by adding some other signals to the coordinates of the functions $View$, $Action$ and $Score$. The current state of each of the automata corresponding to the finite-memory statement will be such a signal.

Another possible expansion of the set of literals is to add an abstract signal. (The abstract signal is determined by an experiment. These abstract signals will be discussed below.) Specifically, we will add the probability of the experiment to return a truth. This probability is a number between 0 and 1. The values of this probability may be one, two, finite and even infinite number. If the possible value is just one (i.e. the probability the experiment returns true is always the same), we will not add such a literal because we are not interested in literals which are constant. If the possible values are infinite number, we can, by rounding them off to some extent, reduce the same to a finite number. It is best if the possible values are two and they are 1 and 0. Then we will have two types of states: those in which the experiment will surely succeed and those in which it will surely not succeed. In the latter case we can make only one experiment to see if the state is of the first or of the second type.

Abstract signals

So far we've talked about the past only. Let's look at the future. With a fixed number of steps, the analogue of statements without memory (the conjunction of past literals) are abstract signals (conjunctions of future literals). The future is a bit more complicated than the past because the past is absolutely determined. With it we know the exact move we've chosen and the information we have received. All this is saved in the history. In the future we have a natural non-determinism. First, we do not know what we'll see, because the function $World$ is non-determined. Second, we do not know what move the machine would choose because it has the right to choose any move it wants. To deal with this non-determinism we will assume that the value of the conjunction is not true or false but true with a certain probability and that with the assumption that the move of the machine is exactly the one it has to be.

The conjunction of future literals will be true with a certain probability on condition that the moves of the machine are the ones they have to be (i.e. all literals associated with the *Action* are true). We can assume that each abstract signal is equal to the probability of a particular experiment to return true.

In the example with the Tic-Tac-Toe game, the following implication is true in the left column of the board:

$horizontal(t) = left \quad \& \quad vertical(t) = nowhere \quad \& \quad put\_cross(t) = no \Rightarrow bad\_move$

That is, in this column the next conjunction is impossible:

$horizontal(t) = left \quad \& \quad vertical(t) = nowhere \quad \& \quad put\_cross(t) = no$

In all other columns the conjunction is possible with a probability of one. That is, if the move of the machine is the one it has to be, the move will be correct and the other will also be fulfilled (actually, there is no other because all literals are associated with *Action*).

This conjunction fully determines the move of the machine, but it is too long (three literals). Let's look at the conjunction with one literal:

$horizontal(t) = left$

This conjunction is impossible in the left column. In the other two columns it is possible with a probability that we can not calculate because we do not know for sure the move the machine will play. For example, the machine can play left and up and the move would be incorrect, not because you cannot move to the left but because you cannot move upwards.

Conjunctions of future literals have another drawback to the conjunctions of past literals. We know the value with the past literals, but do not know the value with the future ones, and we have to guess. We could, on the basis of some implication and the information we have obtained from the past, predict the value of the abstract signal. We want, based on these abstract signals, do statistics and later on – draw conclusions. We can easily do statistics with the past literals. We count how many times the result was truth and that's it. With future literals, however, we do not know how many times it was truth. We have not made the required experiment each single time, and even if we did, the fact that the experiment has succeeded once or failed once does not mean that it will succeed each time (or it will fail each time).

Take for an example the experiment 'If you touch the stove, you will get burned.' This is true when the stove is hot. We would like to conclude that if the stove is hot, we can brew coffee. If you do statistics based on the moments when we touched the stove and we got burned, it can be inferred that if the stove is hot, it will hurt our hand, but this is not because the stove is hot but because we have touched it. Therefore, we will collect statistics based not on the moments when we made the required experiment, but on the moments when we can believe, with sufficient reason, that the stove was hot.

That is, we will try to predict with some probability that the stove is hot and on this basis to conclude that we can brew coffee. We will try to also guess when the stove was hot so that we can collect statistics from these moments. Here, the prediction is different, first because we need more confidence so as

not to spoil the statistics, and secondly because we can use later moments. For example, if we touched the stove or someone else touched it. This happened at a later moment and we could not use it to make the necessary conclusion at that moment, but for the statistics it is important that we have come to know it, albeit belatedly.

How would we represent the internal state of the world?

We would like to somehow represent the internal state of the world. We are not interested in unattainable states of the world and therefore we will take care only of those who are attainable, starting from the initial one. Also we are not interested in equivalent states, and if we have two indistinguishable states, we will take only one of them. So we will assume that S consists of only attainable states and that it is factorized by equivalence relation.

As the set S consists of some arbitrary elements which we know nothing about, for us it would be sufficient to find a larger set in which S can be nested injectively.

Would it be sufficient to describe the state, if we take the n+1-tuple created by $View(s_i)$ and $Score(s_i)$. Yes, if the function $View$ is an injective function, it would be sufficient, but worlds in which the machine sees everything are not interesting. More interesting is when the machine sees only part of the world (i.e. when $View$ is not an injective function).

If there are two distinct states s' and s'', these states would be either immediately distinguishable, i.e. the $Score$ and $View$ functions distinguish them, or there is some experiment that distinguishes them. The experiment may be with a fixed number of steps or be related to an algorithm whose results distinguishes the two states.

Here are examples of experiments with a fixed number of steps (i.e. one step):

'If I touch the stove, I will get burned' or 'If I roll the die, a 6 will come up.'

Here is an example of an experiment related to an algorithm:

'If I roll the die consecutively many times, a 5 will come up before a 6.'

Here's the algorithm:

'I'll roll the die until something bigger than 4 comes up, and then if it is 5 then 'Yes', if it is 6 then 'No'.'

We will call an experiment with a fixed number of steps a conjunction of future literals. We've already explained what future and past literal is.

We will get the number of steps of the experiment by taking the biggest k of the conjunction and divide it by 2 and round up the result.

We will call a closed (or completely determined) experiment the one involving all literals of $Action$ (with values of k from 1to the biggest odd number that is smaller or equal to the biggest k of the conjunction). That is, the operation of the machine is completely determined for the steps of the experiment.

We will call an open (or partially determined) experiment any experiment which is not closed, i.e. the operation of the machine is partly determined. We

can say the open experiment is a set of closed ones because the missing literals of *Action* can be added in all possible ways.

For each state, if we hold a closed experiment, we will get a result 'Yes' or 'No' with a certain probability of 'Yes'. If the function $World$ is determined, this probability would be exactly 0 or 1. If the $World$ is not determined, this number will be a number in the range [0, 1]. That is, every closed experiment gives us a function that at each state gives us the probability the experiment results with a 'Yes'. We've already discussed the number of possible values this function could have. It is best if the possible values are only 0 and 1. For example, if you touch the stove, it is either hot and you get burned, or cold and you do not get burned. The worst case is when the possible value is just one. For example, if I roll the die, will a 6 come up? The answer is 'Yes' with probability of 1/6 and this is true for all states of the world. Of course, there may be a world in which the dice may have curves and if you roll curved dice the probability might be greater than 1/6. There may be a world where mascots help and if you wear your favorite mascot the probability might again increase. That is, two states are discernible with by an experiment not only if one of them results with an 'Yes', and the other with a 'No'. They are discernible even if both experiments result with an 'Yes' with a different probability. For example, it does not matter whether we roll straight or curved dice because the likelihood of a 6 to come up is different.

If you think that the function $World$ is determined, the result of the closed experiment will be determined but that won't help us a lot because we will not be able to predict it accurately and we will expect the result to be 'Yes' with a certain probability. That is, with both possible models of the world (determined and non-determined) we will expect a result of the experiment with the same probability because the facts on which our prediction is made are the same.

If we add to the n+1-tuple the probability of success of all experiments, the result would be an infinite row of signals, but this row will describe fully the state of the world with accuracy to equivalence.

There is no way to add all possible abstract signals. Also, it is better to not restrict ourselves to closed experiments. That's why we will add open experiments as well. They are shorter and in some sense – more representative. For the probability of the open experiment will only consider three possible values: 'firm Yes', 'firm No' or 'maybe'. We will assume that they have at least two values, i.e. at least one of them is something firm. When we unite a set of closed experiments we can not say what the probability of the union is because we do not know what is the probability of each of the actions (that depends on the machine and no one could say what it will do). The case in which we can say what that probability will be is when all experiments result with 'firm Yes'. Then their union is 'firm Yes'. Similarly, if all are 'firm No'. In any other case, the value of the union is 'maybe'.

In our example with the Tic-Tac-Toe game, if the eye is positioned in the left column, its move to the left is incorrect (i.e. impossible). That is, the experiment consisting of only one literal returns a 'firm No' when we are in the left column and 'maybe', if we are in the middle or right column.

Why are abstract signals important?

Abstract signals are very important for understanding the world. When you want to jump over a puddle, it is much easier if in the middle of the puddle there is a stone on which you can step. For example, we find a relationship according to which the stove is hot and the dependence, according to which, if the stove is hot, we can brew coffee. Then based on these two steps, we conclude that we can brew coffee. Could we find a direct rule that under those circumstances we could brew coffee? Yes, but that direct rule would be much longer and more difficult to find. Moreover, it would be very difficult to gather statistics for that direct rule because its precondition could almost never happen. It would be much easier if it is in two steps. We do have some rules that tell us that the stove is hot. Each of these rules has gathered enough statistics for itself. We have statistics that tell us that if the stove is hot, we can brew coffee. This is based on all the cases when for some reason we think that the stove was hot. That is, here we have a generalization of many different preconditions. If you do not use such generalization, the time for the machine to learn will be huge (i.e. the number of steps before it learns will be huge). In theory, we can train ourselves without generalizations, but in practice this cannot happen because when the time for learning tends to infinity, the intelligence of the machine tends to zero.

The importance of abstract signals for AI can be compare with the importance of the Alpha-beta pruning for the game of chess. The Min-Max algorithm can work without this trick, but then the depth of the tree reduces twice. This does not make the algorithm twice slower, but let say a million times slower. The situation is similar with abstract signals. Not only the computing time is tremendously increased but also the training time is tremendously increased. You would not recognize a program to be AI if that program will learn to play Tic-Tac-Toe only after a huge number of moves (such as a million or billion).

Multi-agent world

What we've said so far seems enough to understand any world. In theory, this is really enough, but in practice it is not. In the example with the Tic-Tac-Toe game we saw that in the world there is an imaginary opponent who is part of the world. If we want to describe the world with imaginary opponent then the state of the world would have to reflect the state of this imaginary opponent; whether he is distracted, tired or angry. This would make the description of the world too complicated, so we would like to take out this imaginary opponent of the world so that he is no longer part of the $World$ function.

The old version of $World$, which included the imaginary opponent was:

$s_{i+2} = World(s_i, a_{i+1})$

The new version would be:

$s_{i+2} = World_2(s_i, a_{i+1}, opponent_{i+2})$

That is, we will have a new function $World_2$, which will take as an argument the activity of the opponent as well (added to the state of the world and the

activity of the machine). The new function $World_2$ will be quite simple, because it will not describe the behavior of the imaginary opponent but will only mark the cells by X and O on the game board.

The idea of multi-agent representation of the world is that we are not alone in this world. Apart from us there are many other players (agents). These may be people, animals, Gods or computer programs. Where are the actions of these agents manifested in our world? Once someone does something then that something must appear in some way in the world. The relationship between the agents and the world are abstract signals. There is a signal that we can check in some way. We are trying to predict the value of this signal based on the past or we decide that it is arbitrary and happens with a certain probability. Instead, we can assume that this signal is determined by the will of some other player, unknown for us. That other player may have the power to change this signal whenever he wants, but it is better to think that he is restricted by some rules. For example, in the Tic-Tac-Toe game O's appear in the empty cells. We have an experiment by which we can check what happens to the empty cell. (We go there and see what's in it.) You can assume that the imaginary opponent is restricted and can only mark by O, but cannot remove the mark after that.

To understand the multi-agent world we must have a theory about other agents.

The first thing is to unite a group of abstract signals into an agent. For example, in the Tic-Tac-Toe game someone marks O in the upper left corner or marks O in the lower right corner. It would be better to assume that this is the same agent. Conversely, an abstract signal can be caused by the actions of many agents. For example, if your phone rings, it may not be just one person; it may be a call from a lot of different people.

In the theory that we will build there must be a number of different agents and different abstract signals to associate with different agents. For example, Mom that gives us the bottle of milk, or God responsible for everything.

The main thing to understanding other agents is to split them into good and bad, i.e. partners and opponents. This will help us predict their actions. Whether they will try to help us or hinder us. We should not consider this division permanent. Any partner could become an enemy and vice versa. At the heart of our strategy must stand the desire to enter in contact with these other agents and to make conclusion with them in some way. Examples of such contact is when we give bribe to propitiate a clerk or when we make a sacrifice to propitiate a deity.

Another important thing to understanding other agents is to have a theory about who sees what and who knows what. Does the agent have an opportunity to perform a certain action? As we said there are rules. The agent may be currently unavailable or to be unable to act. It is important whether the agent is smart or stupid. We can hope that he will do something, but to know that he is stupid and will not come up with the idea of doing it.

What we've described looks much like the way people think. All this would be just a small talk if we haven't said what the activity of the agent is and it is the changing of an abstract signal.

Centralized planning

Everything we've considered so far works on a decentralized basis. Indeed, the search for relationships and the building of a model of the world can become decentralized, but when the machine wants to do something it will have to plan its moves. Let's take our example with the Tic-Tac-Toe game. The aim is to win, but the interim aim may be to mark by X the upper left corner of the dashboard. To reach this interim aim we must first move the eye to the upper left corner. In other words, we must have a planning module to make us a plan of action. The plan may represent a number of aims that the machine consistently should reach. The plan may be more complex and not a line but a complicated graph because there may be options.

In the Min-Max algorithm there is no centralized planning of actions. There, on every move it is calculated which is the best move for the machine and that move is played. This algorithm plays successfully chess, but can not play the game Go.

Imagine the following scenario. Let's say you want to go around an obstacle. You can go left or you can go right. If you plan, you will choose where to go and will go around. If you do not plan, you can go left, thing again and go right, then left again and right again. Such behavior would be rather strange, but the worse is that if you do not plan your interim aim and if at every move you ponder on the direction the general goal is, it would be too slow and inefficient. Even the Min-Max algorithm does not seek to win, and instead pursues an increase in the value of the current position of the board. That is, an interim aim has been built in, which is easily visible and easy to pursue. The ultimate aim is too far away and is virtually invisible.

Conclusion

This article gives a pretty detailed description of the AI algorithm; however, we do not need an algorithm but a real running program. What is the difference between an algorithm and a program? The difference is like between an architectural design and a real construction building. When we have a good and workable architectural design we can build a building based on it. It can be too much work but this work is more or less technical. Of course, there is the engineering plan between the architectural design and the actual construction because everything must be well considered; otherwise the building will collapse under the pressure of gravity.

Here the description of the individual modules is very rough and unsatisfactory. Like, for example, the module seeking for dependencies with final memory

(i.e., finite-state automata that are adequate to the world). In our case it is important that we know what we are looking for and in theory we can find it with brute force. Yes, but here brute force will not help us because the possible models (finite-state automata) are too many. Here we need a much more intelligent search method. Ideas that are given in this article might be useful for solving the problem, but the creation of this module remains a serious challenge and is not confined to routine programming. Furthermore, this module can also be a standalone application as a program for finding relationships that will be useful in solving other tasks.

In conclusion, perhaps it would be better not to compare this article to the architectural design of the future building of AI, but rather to compare it to a rough sketch, carelessly scribbled on a paper napkin.

References

1. Turing. Alan., *Computing Machinery and Intelligence*. In: Mind LIX (236): 433460, October 1950.
2. Dobrev D., *AI – What is this*. In: PC Magazine – Bulgaria, November'2000, pp.12-13 (www.dobrev.com/AI/definition.html).
3. Dobrev D., *Formal Definition of Artificial Intelligence*. In: International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285 (www.dobrev.com/AI/).
4. Dobrev D., *Comparison between the two definitions of AI*. In: arXiv:1302.0216 [cs.AI], January, 2013.
5. Dobrev D., *Generator of simple implications*. In: www.dobrev.com/AI/app4.html, made in 2001 and published in 2008.
6. Balbiani Philippe, Gasquet Olivier, Schwarzentruber Franois, *Agents that look at one another*. In: Logic Journal of the IGPL (2013), vol. 21 (N. 3). pp. 438-467. ISSN 1367-0751
7. Dochev D., G. Agre, R. Pavlov, *User Authoring in Learning-by-Doing Situations*. In: Rachev B., A. Smrikarov (Eds.) Proc. of CompSysTech 2011, Vienna, June 2011, ACM ICPS Vol. 578, ACM PRESS, ISBN: 978-1-4503-0917-2, pp. 577-582.