Consistency, Meaning, and Machines

--- Achieving a consistency as an intension of a machine,
and Meanings of strings for the machine ---

Nov. 7, 2020

Kenzo Iwama

Key words: Meanings of a sentence, Symbol grounding, Embodiment, Machine intension, Consistency among the input and the retrieved, Prediction as a result of achieving the consistency

1. Introduction

In this note I briefly summarize much research conducted for making a machine acquire meanings of symbols, and explain a way we can take to construct a machine with its intension and having meanings of strings. I make the machine in such a way that it gets input strings through four channels and finds regularities in them. When the machine gets a new string, it retrieves a string of a regularity that matches the input before the new input string ends. The new input string continues and the retrieved string also continues. The machine sees if the continuing input string matches the continuing retrieved string. If it sees the matching, namely the inputs and the retrieved are consistent to each other, it is said that the machine becomes able to predict how the string continues, and the regularity found describes the world outside the machine.

The intension of the machine is to make predictions, and to do so, the machine tries to find regularities in input strings and to achieve the consistency between a new input string and a string kept as the regularity. In general, the more input strings the machine gets and keeps in its memory, the more reliable regularities the machine is able to form. As a result of forming the more reliable regularities, the machine is able to make a more accurate prediction. The meaning of a string is an increase in the probability with which the machine, given a new string, accurately predicts how the new string continues. When the machine has the meaning of the string (the machine accurately predicts the string continues), it becomes to say, given a new string, how the new string would continue if the new string stops continuing.

2. Developing a machine that has meanings of strings

2.1 Rules, or syntax that a current machine follows

It is true that a current machine on which a program runs has an initial state, and changes states from one state to the next state until a final state. Or the machine with the program may run forever. Any change from one state to the next state is determined at the time the program starts to run. A sequence of states from the initial state to the final state may be different depending on input data after the program gets, but the dependency is also determined at the time the program starts to run.

It is also true that a current machine with a program is "… by definition a device that manipulates formal symbols. These are usually described as 0s and 1s, though any old symbol will do as just as well. … Computation, so defined, is a purely syntactical set of operations, in the sense that the only features of the symbols that matter for the implementation of the program are the formal or syntactical features".

Formal symbol

When a programmer writes a program, he / she uses symbols (letters, words, and sequences of words) so that

1) the programmer allocates areas in a program space and specifies operations on values stored in the areas.
2) the programmer specifies functions and variables in such a way that each specified function takes which and which variables as its function variables.
3) the programmer specifies logical relations and variables in such a way that each logical relation holds among which and which variables.

In any case the programmer specifies what relation holds among which and which variables by using the following two rules: 1) the order of letters decides a word, and 2) syntax of words decides the order of operations, inputs and outputs of each function, and what logical relation holds among variables.

2.2 Meaning of symbols – Chinese room argument

Stanford Encyclopedia of Philosophy summarizes the Chinese room argument by Searle as follows: "Searle imagines himself alone in a room following a computer program for responding to Chinese characters slipped under the door. Searle understands nothing of Chinese, and yet, by following the program for manipulating symbols and numerals just as a computer does, he sends appropriate strings of Chinese characters back out under the door, and this leads those outside to mistakenly suppose there is a Chinese speaker in the room".

The summary continues "The narrow conclusion of the argument is that programming a digital computer may make it appear to understand language but could not produce real understanding. Hence the "Turing Test" is inadequate. Searle argues that the thought experiment underscores the fact that computers merely use syntactic rules to manipulate symbol strings, but have no understanding of meaning or semantics. The broader conclusion of the argument is that the theory that human minds are computer-like computational or information processing systems is refuted. Instead minds must result from biological processes; computers can at best simulate these biological processes.”

For example, a computer, given a string “五足す五”, invokes a program (or sees an instruction) that tells consulting a Chinese-English dictionary by comparing a string “五”, “五足す”, or “五足す五” and words in the dictionary to see if the dictionary has it. “五” is in the dictionary. The program (the instruction) tells extracting “5”. Next, the computer again invokes the program that tells comparing a string “足す” and words in the dictionary. The computer extracts “plus”. Then the program tells “5 plus 5” is an English sentence corresponding to the string “五足す五”, and outputs it. Whenever the computer, given a Chinese string, outputs an English string that is the same as a human translator, given the same Chinese string, writes. Then the computer appears to understand Chinese and English as the human translator. However, the computer does not understand Chinese or English after it gets Chinese strings and then outputs English strings many times.

To make a machine understand Chinese sentences, I construct the machine in such a way that the machine has its original program and the original program acquires firstly English sentences with their meanings and secondly how to translate Chinese sentences into English. For acquiring the sentences with their meanings, it binds inputs through four channels; inputs through one channel are English sentences in quotations, inputs through the second and the third channel are strings made by sensor and motor devices, and inputs through the forth channel describe internal states of the program. For example, a meaning of strings, “there are 5 ○” ｛○○○○○｝, is that the machine, given the strings, confirm 5○ in ｛○○○○○｝. The machine firstly acquires a way of seeing 5○ in ｛○○○○○｝, and binds the way and “there are 5○” together. Then the machine acquires “五” pointing to the acquired. (I claim the strings have other meanings than just described, and explain them later in the note.)

2.3 Meaning of a sentence

A procedure as the meaning of a sentence

Several approaches are taken toward developing a machine that understands the meaning of a sentence. It seems to me that Gierasimczuk (2007) gives us a good summary of one approach to investigate into the meaning of a sentence. It says “According to an old philosophical idea, the meaning of a natural language construction can be identified with a representation of its denotation [Frege 1892]. This thought has been developed in the direction of identifying the meaning of an expression with a procedure for finding its extension [Tichy 1969, Moschovakis 1990, van Lambalgen, Hamm 2004, Szymanik 2004]. In the case of words: the meaning of “Poland” is the procedure of checking if the object in question satisfies conditions of being Poland. In the case of sentences: the meaning is a procedure for finding a sentence’s logical value. The meaning of “Alice has a cat.” is a procedure for checking if Alice really has a cat. Therefore, we can say that someone understands a sentence (knows its meaning), if he knows a procedure for checking whether it is true or not”.

I think Gierasimczuk and others are right if 1) the meaning of a sentence is limited in a narrow scope and 2) the procedure is assumed to be developed. Limitations they place are the following; 1-1) An intention of a speaker as well as a listener is out of their scope. When a speaker gives a sentence to a listener, the speaker has the intention of giving the sentence, and the listener, getting the sentence, has the intention to satisfy the intention of the speaker (or do something else). The listener, given the sentence Ss, tries to retrieve the procedure to evaluate the sentence. But it often happens the listener does not retrieve the right procedure, and tries finding / devising a procedure Pn to get the logical value of the sentence Ss. Trying to get the procedure can be the meaning of the sentence Ss for the listener (and may be for the speaker). It also happens the listener retrieves the procedure and evaluates the sentence Ss by performing the procedure, but gets the logical value different from that by the speaker (and others). This case can also become the meaning of the sentence Ss for the listener. 1-2) Binding a sentence and sensorimotor features (or states) together (or grounding a sentence to its environment) is out of their scope. (I believe getting and putting features (or strings, words, sentences) through several channels, and binding them together are inevitable to make the sentence have its meaning. In the next section, I discuss symbol grounding problem and binding strings together.)

For lifting the limitation described under item number 1-1) in the above, I think it desired to extend the process of evaluating a sentence by integrating the following; 1-1) collect sentences and procedures each of which evaluates the logical value of each sentence and 1-2) devise a procedure Pn out of the collected sentences and the procedures. 1-3) keeping the difference between the evaluation result by the listener and that by the speaker with the following: the sentence, objects neighboring to the sentence, and the procedure the listener uses. Later, try to achieve the consistency between the result by the speaker and that by the listener (in other words, resolving the difference).

For example, suppose a child is given a sentence “2 and 3 is the same as 5”, the child may not have a procedure to see if the sentence is true. Then the child may take sentences with procedures out of his / her memory, and combines the procedures to evaluate the sentence; Put as many blocks as asked, such as “put 2 blocks; put one and put the second one on top of the other” and “put 3 blocks; put one, put the second one on top of the first one, and put the third one on top of the two blocks”. Put blocks on top of other blocks already placed such as “put 2 blocks on top of 3 blocks already placed”. “See the height of blocks put is the same as that of blocks put”. For the child, getting such procedures contributes to devising a way to see if “2 and 3 is the same as 5”.

The same as the above can be said for a machine; the meaning of strings is a procedure that the machine evaluates the logical value of the strings. When the machine does not retrieve the procedure to evaluate the logical value, the meaning of getting the strings become collecting procedures (bunch of strings) so that the machine devises a new procedure out of the procedures to see if the given strings are true or not. When the logical value is different from that by other machine, it is meaningful for the machine to get the sentence because the machine keeps the sentence, objects neighboring the sentence, and the procedure, and later it tries to see why the result is different from that by the other, and to develop a procedure to cover the two cases.

Intentional logic (or other formal system)

Some researchers take an approach for representing the meaning of a sentence more than its designation, and study intentional logic. Fitting (2020) explains motivation behind the study of intentional logic “There is an obvious difference between what a term designates and what it means. At least it is obvious that there is a difference. In some way, meaning determines designation, but is not synonymous with it. After all, “the morning star” and “the evening star” both designate the planet Venus, but don’t have the same meaning. Intentional logic attempts to study both designation and meaning and investigate the relationships between them.”

Fitting continues to say “Contexts in which extension is all that matters are, naturally, called extensional, while contexts in which extension is not enough are intentional Mathematics is typically extensional throughout—we happily write "1+4=2+3" even though the two terms involved may differ in meaning. … In classical first-order logic intension plays no role. It is extensional by design since primarily it evolved to model the reasoning needed in mathematics. Formalizing aspects of natural language or everyday reasoning needs something richer. Formal systems in which intentional features can be represented are generally referred to as intentional logics."

It seems the study of intentional logic shows that 1) one finds cases where the one is not able to represent the meaning in a formal system already developed while the one studies more of the meaning of a sentence than before and 2) the one sees the necessity to refine / revise the formal system and represents the meaning in the refined / revised system. Although a formal system in general does not allow the one to describe sensorimotor features in the meaning of a sentence since logical symbols are not bound to the sensorimotor features as explained in the next section, the one becomes able to explicitly describe more of the meaning in the refined system than before.

The fact that the one finds cases where the one is not able to represent the meaning of a sentence in a formal system implies the one finds the difference between the represented and the meaning of the sentence (or some in the meaning is missing in the represented). The difference is seen (or detected) when some phenomena (or features / aspects) recalled (or remembered) by the sentence cannot be mapped to the represented. In other words, the one finds inconsistency between the represented and the phenomena recalled (or carried) by the sentence.

Then I think it is required to make the machine find (or detect) the missing in the represented (or inconsistency between the input and the represented) and have a method to fulfill the missing (or resolve the inconsistency) in order to construct a machine that understands the meaning of a sentence.

What I have been trying to is make the machine form hierarchical constructs / regularities when the machine finds inconsistency between the input and the retrieved. The constructs describe both cases where the input are not mapped to the retrieved and cases where the input are mapped to the retrieved, and the constructs include a way from the former cases to the latter cases.

2.4 Meanings of symbols – Symbol grounding problem

Symbol grounding problem has been seriously studied for the purpose of developing a machine that understands the meanings of a sentence and behaves intelligently. Taddeo and Floridi (2005) briefly states the problem as follows; "The symbols are merely a part of a formal, notational convention agreed upon by its users. One may then wonder whether an artificial agent, such as a robot, may ever be able to develop an autonomous, semantic capacity to connect its symbols with the environment in which the artificial agent is embedded interactively."

When Harnad (1990) introduced the problem, he asked "How can the semantic interpretation of a formal symbol system be made intrinsic to the system, rather than just parasitic on the meanings in our heads? How can the meanings of the meaningless symbol tokens, manipulated solely on the basis of their (arbitrary) shapes, be grounded in anything but other meaningless symbols?" Cangelosi and Harnad (2001) continue describing the problem; "The symbols, in other words, need to be connected directly to (i.e., grounded in) their referents; the connection must not be dependent only on the connections made by the brains of external interpreters like us. Just the symbol system alone, without this capacity for direct grounding, is not a viable candidate for being whatever it is that is really going on in our brains when we think meaningful thoughts."

Reviewing much work toward solving the symbol grounding problem, Taddeo and Floridi (2005) state the requirements the solution must satisfy; "the interpretation of the symbols must be intrinsic to the symbol system itself, it cannot be extrinsic, that is, parasitic on the fact that the symbols have meaning for, or are provided by, an interpreter." and put forth two conditions: "a) no form of innatism is allowed; no semantic resources (some virtus semantica) should be presupposed as already pre-installed in the artificial agent; and b) no form of externalism is allowed either; no semantic resources should be uploaded from the "outside" by some deus ex machina already semantically-proficient." They continue to state "Of course, points (a)-(b) do not exclude the possibility that c) the artificial agent should have its own capacities and resources (e.g. computational, syntactical, procedural, perceptual, educational etc., exploited through algorithms, sensors, actuators etc.) to be able to ground its symbols." They review eight strategies proposed for the solution of the SGP in the last fifteen years, and say the conclusion is that none of them satisfies the conditions put forth in the above.

The problem is still open as of 2015 (Bielecka 2015, Bringsjord 2015). It seems that assuming / considering the existence of two separate worlds makes the problem difficult; one is a formal symbol system and the other is the environment (including interactions between the symbol system and the environment). The formal symbol system is one of the latest results in the history of developing concepts / thoughts. Words / sentences seem to have emerged to share something (or phenomena) in one's brain with that in another one's brain at the very early human history. If we try to get hints from language acquisition by children to guess how the emergence occurs, it seems improper to me to assume the two separate worlds. During some development stage an infant utters meaningful voices when the infant does something on objects and the infant apparently has acquired basic behaviors of objects (such as roll, fall, and stay) before the infant uses words describing the behaviors of the objects. The infant appears to share how the objects behave with its parent and to have the parent do something by uttering voices so that the objects behave as the infant knows (or images). Another phenomena we can get hints to guess the emergence of sentences is that a child raises questions such as "what is this", "where is a candy", after the child acquires various words with their meanings. Those questions seem to consist of relations between objects the child is dealing with and the words already acquired by the child.

Therefore, it seems reasonable to develop a machine that have words with their meanings by introducing a stage where the machine makes utterances to share something occurring inside the machine with another machine or a human before the machine forms words / sentences to mean something. I think a proper way to investigate is how a machine forms concepts, in a parallel and layered way, that integrate the environment, the interactions with the environment, and audio inputs / verbal outputs (or sequences of audio and verbal features).

I try constructing a machine that binds features extracted and serialized in the temporal order through several channels, and finds regularities in them, assuming the features are extracted and serialized. When the machine finds the regularities, it tries to have another machine do the same by using utterances. If the utterances keep to be the same for one regularity, the correspondence between the utterances and the regularity is kept as a regularity and the utterances become meaningful voices (and then words). Moreover the machine makes hierarchical regularities that cover both cases that follow regularities already formed and cases that do not follow the regularities. An example of a hierarchical regularity is the one that includes auditory / verbal features, "where is a candy”. When the machine is asked to do something on some object by someone, the object is usually within a visual scope of the machine. But there are cases where the object is not in the visual scope. Then the machine forms a regularity to look for the object and/or asks the one using audio features “where is the object”. (A motivation to look for the object and/or to raise the question is caused by the intension of the machine, which I discuss later in this note.)

After the machine has acquired how objects behave when the machine moves the objects and sentences to describe the motions and behaviors, the machine acquires the number system. It firstly gets and puts features though several channels in the form of strings such as “4 ○” ｛ ○○○○ ｝. “4” is auditory feature in the first place and then becomes a member of the number. (I describe how the machine devises the number system, in a separate paper, to achieve consistency concerning repetitions of motions (or actions) between the machine at one time and that at a later time and between the machine and another machine.

A problem of seeing features in continuing inputs (and / or extracting features from the continuing inputs) is open. Researchers in robotics, AI, and others have been making much effort to solve the problem.

2.5 Embodiment

Embodiment has been considered to be a key to make an intelligent robot, and symbols were expected to be connected to the meanings through interacting with the environment if any symbols are used by the embodied robot. For example, Pfeifer and Scheier (1999) say “The problem of embodiment refers to the fact that abstract algorithms do not interact with the real world. Rodney Brooks forcefully argued that intelligence requires a body (Brooks 1991a, 1991b). Only if a system is embodied we know for sure that it is able to deal with the real world. Moreover, systems that are not embodied all suffer from the symbol grounding problem.”

Ziemke (2016) reviews research activities to make a system embodied; “Embodied approaches to AI – using robotic or simulated ‘autonomous agents’ – at least at a first glance, allow computer programs and the representations they are using, if any, to be grounded in interactions with the physical environment through the robot/agent platform’s sensorimotor capacities. Brooks, for example, one of the pioneers of embodied

AI, formulated what he called “the two cornerstones of the new approach to Artificial Intelligence, situatedness and embodiment” (Brooks, 1991). Embodiment from this perspective simply means that “robots have bodies and experience the world directly – their actions are part of a dynamic with the world and have immediate feedback on their own sensations” (Brooks, 1991). According to Brooks, such systems are physically grounded, and hence internally “everything is grounded in primitive sensor motor patterns of activation” (Brooks, 1993). Situatedness, accordingly, means that “robots are situated in the world – they do not deal with abstract descriptions, but with the here and now of the world directly influencing the behavior of the system” (Brooks, 1991).”

Although much effort has been made to develop an embodied robot and to see how the robot have symbols be grounded and be meaningful, Pfeifer and Iida (2003) describe the state of research as of the year 2003; “One of the big unresolved issues to date is the one of symbol processing: How is it possible that humans have the capability for symbol processing? More precisely we would have to ask how it is possible that humans can behave in ways that it makes sense to describe their behavior as “symbolic”, irrespective of the underlying mechanisms, which might involve explicit symbol processing or not. The question is very broad and of general importance: it is about how organisms can acquire meaning, how they can learn about the real world, and how they can combine what they have learned to generate symbolic behavior, a problem known as the “symbol grounding problem.”. There is general agreement that learning will make substantial contributions towards a solution. However, learning alone will not suffice – embodiment must be taken into account as well.”

Ziemke (2016), siting from the work by Chemero, describes embodied cognitive science have not resolved the issue as stated by Pfeifer and Iida (2003); “Chemero’s (2009) characterization of the current embodied cognition research landscape, that there currently are at least two very different positions/traditions that are both referred to as ‘embodied cognitive science’. One of these, which Chemero refers to as radical embodied cognitive science, is grounded in the anti-representationalist and anti-computationalist traditions of eliminativism, American naturalism, and Gibsonianecological psychology. The other, more mainstream version of embodied cognitive science, on the other hand, in line with what was referred to as robotic functionalism above, is derived from traditional representationalist and computationalist theoretical frameworks, and therefore also still is more or less compatible with these – as illustrated maybe most prominently by the notion of symbol/representation grounding, as opposed to the more radical position of anti-representationalism.

The position of radical embodied cognition, according to Chemero (2009), can be summarized in two positive claims and one negative one:
1. Representational and computational views of embodied cognition are wrong.
2. Embodied cognition should be explained using a particular set of tools T, including dynamical systems theory.
3. The explanatory tools in set T do not posit mental representations."

Facing the difficulty to construct a robot that has representations / models of its environment be grounded and / or use symbols that have meanings, some researchers (Johnson 2007, Ziemke 2016), referring to works in synthetic biology, propose an approach to overcome the difficulty; For example Johnson (2007) says "In retrospect I now see that the structural aspects of our bodily interactions with our environments upon which I was focusing were themselves dependent on even more submerged dimensions of bodily understanding. It was an important step to probe below concepts, propositions, and sentences into the sensorimotor processes by which we understand our world, but what is now needed is a far deeper exploration into the qualities, feelings, emotions, and bodily processes that make meaning possible.", and Ziemke (2016) says "modeling organisms as layered networks of bodily self-regulation mechanisms can make significant contributions to our scientific understanding of embodied cognition."

It seems that the difficulty remains even if research into sensorimotor processes to construct models of self-regulation mechanisms is conducted because the symbols remain separated from the models of the body unless a way of how symbols emerge out of something (possibly units made out of sensorimotor mechanisms) is incorporated into bodily self-regulation mechanisms.

I think it possible to overcome the difficulty. A way is to construct a machine that gets continuing inputs and finds regularities in the inputs to form the regularities, and tries to keep consistency among the inputs and the regularities formed. But cases occur where some inputs are not consistent to regularities formed and then the machine forms hierarchical regularities to achieve consistency among the regularities and the inputs. The machine incorporates features of audio signals in regularities and then the audio features become meaningful sentences. Here the machine is not a computer but it has states and changes them from one to the next. Importantly it has no symbols, from the beginning, that need to be grounded. A human writes a program running on the machine, and programming constructs such as constants, variables, instructions, are for specifying states and changes of the states. Each state consists of beginning, intermediate, and end of actions such as getting inputs, forming regularities, retrieving regularities formed.

2.6 Achieving consistency as an intension

Intension of a robot has become got much attention by people since robots are introduced in various fields such as care robots and autonomous cars. A human could do his / her role with ease in the fields when he / she is able to expect what to do and how to behave. The human tend to expect what and how the robot will do next when he / she sees actions, verbal outputs, and/or some deed by the robot. Therefore it becomes important to show to the human what and how the robot is going to do. Then the human regards the robot to have its intension.

A robot studied / developed in such fields shows its intension that consists of values produced by a set of functions defined (or procedures or logical sentences written) by a human. Input values to the functions are determined by the environment. The intension shown by the robot appears to be intrinsic as long as the input values are within the domain of the functions because the robot, given the input values, gets output values of the functions which show future behaviors of the robot. (In practice, the range the robot moves around is limited into the environment where function values are kept within the domain of the function.)

It seems that intension of a machine not just appears intrinsic but becomes really intrinsic when the machine gets values that are not within the domain of the functions and the machine is able to devise a method to deal with such values. In general the machine is considered to have its intension when it has some goal and 1) it does a way to the goal if it has the way 2) it devises a way to the goal if it does not have the way. Devising the way becomes a new goal for the machine to achieve and may be carried out 1) by firstly seeing the machine does not have the way, 2) by secondly setting a new goal (explicitly or implicitly) to find / form the way: revising functions that the machine already has, forming a new function (or a method) out of functions (or methods) that the machine has, or employing some other way to achieve the new goal, and 3) by executing the way found.

I have been trying to construct a machine that has its intrinsic intension although the intension is very limited. A human writes a machine program, and then the program tries to have its intension become true; namely the program predicts, given a new string (namely sensorimotor features), how the new string continues. The intension is satisfied when the predicted strings are consistent to the new continuing strings.

For example, after the machine gets various inputs and forms regularities, the machine gets the following inputs (an instance of a math problem); "There are two numbers. Addition of the two numbers is 100. The difference of the two numbers is 10. Find the two numbers." And gets the following (an instance of the way to solve the math problem); "Subtract 10 from 100 is 90. Divide 90 by 2 is 45. Add 10 to 45 is 55. One number is 45. The other number is 55." Then the machine gets a new instance of the math problem, "There are two numbers. Addition of the two numbers is 210. The difference of the two numbers is 18. Find the two numbers." The machine makes a prediction (or solves the problem); "Subtract 18 from 210 is 192. Divide 192 by 2 is 96. Add 18 to 96 is 104. One number is 96. The other number is 104." The strings predicted become consistent to the continuing input strings.

A hierarchical regularity as a means to achieve consistency

Various (autonomous) mechanisms may construct systems that achieve (or keep) consistency among two things that are not initially consistent with each other. Constructing a hierarchical regularity is a result of achieving consistency among the two things; one is cases where a regularity already formed holds and the other is cases where the regularity does not hold. The hierarchical regularity integrates the two cases. For example, suppose a regularity that objects are bound to audio features (or strings) is formed. But cases where objects are not bound to strings occur, and are not consistent to the regularity. Then the two cases are integrated to form a hierarchical regularity, and the regularity is bound to audio features "what is this" or "what is the name of this". For another example, suppose a regularity an object is within a visual / other sensor's scope when audio features specify to do something on the object is formed. But cases where an object is not within a visual / sensor's scope occur. Then the two cases is integrated to form a hierarchical regularity. Audio features "where is the object" are bound to the hierarchical regularity.

3. A machine and a computer

3.1 The size of grain, a set of values, and a series of steps

The size of grain matters; People have a concept of a tube of a bicycle, and takes it as a whole grain. People have also a concept of molecules of tube rubber, and considers the molecule to be a whole grain. They have the two concepts separately, and takes relations between them into their consideration if necessary.

People have a concept of a computational step and regards it as a single manipulation when they develop computer programs and run them on a computer. But a designer of a compiler (a program that translates a program written in a programming language into a program in a machine language) looks inside a computational step, and sees each computational step consists of several machine steps (a machine step is specified by a machine language). An electronic engineer sees electronic circuits; for example, he designs an adder which is an electronic circuit specialized for adding two numbers in digits. The adder consists of logic gates. Each gate in the adder performs logical operations on two digits such as "1 and 1 is 1", "1 and 0 is 0", or "1 or 0 is 1". An engineer regards a logic gate as a whole grain.

Similarly a set of values in a memory space can be considered as a state (or a pattern), and the set of the values can be mapped to a state outside of a machine. Each value in the set is manipulated by program steps, but a set of values can be considered to be taken or put by one action of the machine. For example, a string of audio signals (the set of values) is compared to various strings of audio signals (sets of values) stored in a memory by an action of comparing two audio signals although the action uses a sequence of program steps.

For another example, a bundle of values (or a string) "Find the number." becomes a target of comparison, and the bundle is compared to many bundles stored in a memory. When the machine finds the matched bundle, it retrieves the bundle with bundles following, and executes the bundle and then the bundles. In other words, the machine invokes a sequence of mathematical steps such as "Subtract 18 from 210 is 192." One mathematical step is considered a single manipulation, but each mathematical step consists of many program steps. When people place their focus on mathematical steps, they tries to trace mathematically correct steps. On the other hand, they sees program codes to execute one mathematical step when they place their focus on how the codes written by a programming language conducts the mathematical step.

3.2 A machine conducts more than a so-called computer

A machine on which I have written and run a program is a so-called computer. But the machine I have been trying to construct has the following features: 1) It has states each of which can be mapped to an instance of the interaction between the machine and the environment. Changes from one state to the next can also be mapped to changes of the interaction. Here one state consists of many values stored in memory of the machine, and one change is performed by many programming steps. When one takes the size of grain into consideration, many values can become one state, and many steps can be one change. For example, counting the number of objects consists of a sequence of counting steps, the machine got the sequence and kept the sequence for later use. Each counting step can be mapped to a state inside the machine. 2) It has its intension to make input strings and retrieved strings be consistent. For example, suppose it has formed sequences of counting the number of objects and has kept them in its memory. Suppose the machine is given the beginning part of the counting sequence and is not given the latter part. Then it retrieves the sequence of counting and continues counting by using the retrieved as if the latter part keeps continuing and the latter part is consistent to the retrieved. 3) It has an ability of forming (or devising) a new function such as solving a new mathematics problem, and of forming a new set whose member is defined by the new function.

The feature 3) in the above cannot be considered as any of the characteristics of recursive functions since a new function cannot be an output of the recursive functions. Therefore the machine is not a computer in the sense that the computer computes the recursive functions. I think that Church's and Turing's thesis does not hold for a machine with a program I have been developing if one takes the size of grain into consideration. The machine I have been constructing does wider than what a so-called computer does.

Moreover it seems what Turing said about computation by a human is limited to performances made using the formulated; "Turing argued that, given his various assumptions about human computers, the work of any human computer can be taken over by a Turing machine. Whatever sequence the human computer is computing, a Turing machine "can be constructed to compute the same sequence" (Copeland 2020). But it might be proper what a human computes include making his / her intension become true; in particular, making what he / she computes be consistent to what other human computes. The human sometimes tries to create functions (or methods) and to get results; for example, a student, given a problem "There are two numbers. Adding them together makes 132. The difference between the two is 14. Find the two numbers", constructs (or devises) a method out of methods already the student has acquired and finds results even though the student did not get any instances of solving the problem or any solutions to the problem. The student may form a new set of instances of the problem in the brain. In the sense as just described, a currently available computer does not compute as a human computes.

3.3 Meaning of a sentence

When the machine gets a sentence, it retrieves sentences already got, from its memory, and it uses them to predict what sentences it gets continuously (or how the environment and the interaction continue) after the sentence. In general, the more sentences the machine gets, the more reliably the machine makes a prediction about the environment and the interaction. When the machine gets a sentence *S*, the machine makes an accurate prediction with the increased probability. The meaning of the sentence *S* for the machine is the increase in the probability.

The meaning defined here includes the meaning discussed in Section 2.3. "The meaning of a sentence" is proposed in Section 2.3 to be "a procedure that evaluates its logical value." The cases discussed there form a subset of the cases described in the above paragraph, namely, the cases where the machine gets the sentence, retrieves sentences and uses them to predict (or just get) what sentences it gets continuously. If the prediction turns out to be correct (or consistent to the environment and the interaction), the logical value is true. Otherwise, the logical value is false.

4. Discussion

Strings as features

Although much effort is needed to find and form features in continuing inputs, I assume the features are formed / extracted, and a machine gets and puts the features as its inputs and outputs. In my description of a machine, I replace features by strings to specify inputs to and outputs from the machine. The machine gets strings through several channels and binds them together when it gets them at the same time. The machine finds regularities in continuous streams of strings. The meaning of strings are with the regularities, and in particular, the designation is made by the binding.

Infinity

I do not discuss how a machine becomes to deal with the infinity or devises a way of keeping consistency among numbers including infinite numbers. I will describe how the machine finds the infinity and integrates it into mathematical problems in a new paper.

Grammars and most general sets

I need to make the following clearer than just described. Orders of words, grammars, are related to how audio strings are taken out of hierarchical regularities that keep the audio strings. The orders of the audio strings are determined so that the orders of a speaker and those of a listener become the same, and are kept in a regularity. While the orders become the same among various words, the orders of representatives of most general sets, objects and actions, become to decide the orders of members of the sets. Since the hierarchical regularities include regularities that include objects and actions on the objects, taking out of audio strings from the hierarchical regularities become a cause of production rules of words.

Inductive program synthesis

Kitzelmann (2009) says, in Inductive Programming: A Survey of Program Synthesis Techniques, "In classical software engineering and algorithm design, a deductive—reasoning from the general to the specific—view of software development is predominant. One aspires a general problem description as starting point from which a program or algorithm is developed as a particular solution." and continues "Inductive programming, on the other side, aims at developing methods based on inductive—from the specific to the general—reasoning (not to be confused with mathematical or structural induction) to assist in programming, algorithm design, and the development of software. Starting point for IP methods is specific data of a problem—use cases, test cases, desirable (and undesirable) behavior of a software, input/output examples (I/O-examples) of a function or a module interface, computation traces of a program for particular inputs and so forth. Such descriptions of a problem are known to be incomplete. Inductive methods produce a generalization of such an incomplete specification by identifying general patterns in the data. The result might be again a—more complete—specification or an actual implementation of a function, a module, or (other parts of) a program."

Researchers in inductive programming makes a summary, in Approaches and Applications of Inductive Programming (2019), as "Inductive program synthesis is of interest for researchers in artificial intelligence since the late sixties (Biermann and others 1984). On the one hand, the complex intellectual cognitive processes involved in producing program code which satisfies some specification are investigated, on the other hand methodologies and techniques for automating parts of the program development process are explored. One of the most relevant areas of application of inductive programming techniques is end-user programming (Cypher 1993, Lieberman 2001, Cypher and others 2010)."

I think methods / techniques developed in the field of inductive program synthesis may be incorporated into a machine I have been constructing. In particular, it is desirable to extract criteria / metric from their work to find / identify a regularity in many examples, and to sort out examples into classes / categories consisting of the examples.

Bibliography

Bielecka, K. (2015). Why Taddeo and Floridi did not solve the symbol grounding problem. J. Experimental & Theoretical Artificial Intelligence, 27, 79-93.

Biermann, A. W., Guiho, G. and Kodratoff, Y. (Eds.) (1984). Automatic Program Construction Techniques. Macmillan.

Bringsjord, S. (2015). The symbol grounding problem … remains unsolved. J. Experimental & Theoretical Artificial Intelligence, 27, 63-72.

Brooks, R. A. (1991). Intelligence without representation. Artificial Intelligence, 47, 139-159

Brooks, R. A. (1991). Intelligence without reason. Proceedings of the twelfth international joint conf. on artificial intelligence. Morgan Kauffman, 569-595.

Brooks, R. A. (1993). The engineering of physical grounding. Proceedings of The Fifteenth Annual Conference of the Cognitive Science Society, 153-154. Lawrence Erlbaum Associates.

Brooks, R. A. and Stein, L. A. (1993). Building brains for bodies. MIT Artificial Intelligence Laboratory Memo 1439.

Cangelosi A. and Harnad S. (2001). The adaptive advantage of symbolic theft over sensorimotor toil: Grounding language perceptual categories. In L. Steels (Ed.) The evolution of grounded communication, Special issue of evolution of communication, 4:1, 117-142. J. Benjamins Publ.

Chemero, A. (2009). Radical embodied cognitive science. MIT Press.

Copeland, B. J. (as of Summer 2020). The Church-Turing Thesis. In E. N. Zalta (Ed.) The Stanford Encyclopedia of Philosophy. URL = <https://plato.stanford.edu/archives/sum2020/entries/church-turing/>

Cypher, A. (Ed.) (1993) Watch What I Do: Programming by Demonstration. MIT Press.

Cypher, A., Dontcheva, M., Lau T. and Nichols, J. (Eds.) (2010). No Code Required: Giving Users Tools to Transform the Web. Elsevier.

Dąbrowska, E. and Lieven, E. (2005). Towards a lexically specific grammar of children's question constructions. Cogn. Linguist. 16, 437–474.

David, C. (as of Spring 2020). The Chinese room argument. In E. N. Zalta (Ed.) The Stanford Encyclopedia of Philosophy. URL=<https://plato.stanford.edu/entries/chinese-room/>.

Fitting, M. (as of Spring 2020). Intensional Logic. In N. Zalta (Ed.) The Stanford Encyclopedia of Philosophy. URL = <https://plato.stanford.edu/archives/spr2020/entries/logic-intensional/>.

Frege, G. (1892). Über Sinn und Bedeutung,Zeitschrift für Philosophie und philosophische Kritik 100, 25-50.

Guasti, M. T. (2002). Language Acquisition: The Growth of Grammar. MIT Press.

Gierasimczuk, N. (2007). The problem of learning the semantics of quantifiers. In B. D. ten Cate and H. W. Zeevat. (Eds.) Logic, Language, and Computation. LNCS, volume 4363, 117-126. Springer.

Harnad, S. (1990). The symbol grounding problem. Phys. D. 42, 335-346.

Huttenlocher, J., Vasilyeva, M., Cymerman, E., and Levine, S. (2002). Language input and child syntax. Cognit. Psychol. 45, 337–374.

Johnson, M. (2007). The meaning of the body: Aesthetics of human understanding. U. of Chicago Press.

Kitzelmann, E. (2010). Inductive Programming: A Survey of Program Synthesis Techniques. In U. Schmid, E. Kitzelmann, and R. Plasmeijer (Eds.) Approaches and Applications of Inductive Programming. LNCS, volume 5812, 50-73. Springer.

Van Lambalgen, M. and Hamm, F. (2004). Moschovakis' Notion of Meaning as applied to Linguistics. In M. Baaz, S. Friedman, and J. Krajicek (Eds.) Logic Colloquium '01. Lecture Notes in Logic, 255-280. Cambridge U. Press.

Lieberman, H. (Ed.) (2001). Your Wish is My Command: Programming by Example. Morgan Kaufmann.

Lloyd, J. W. (1984). Foundations of Logic Programming. Springer.

Mitchell, T. M. (1997). Machine Learning. McGraw Hill Comp.

Morse, A., Herrera, C., Clowes,R., Montebelli, A. and Ziemke. T. (2011). The role of robotic modelling in cognitive science. New Ideas in Psycol. 29(3), 312-324.

Moschovakis, Y. (1994). Sense and denotation as algorithm and value. In J.Oikkonen and J Väänänen (Eds.) Logic Colloquium '90. Lecture Notes in Logic, 210-249. Cambridge U. Press.

Muggleton, S. (1991). Inductive Logic Programming. New Generation Computing, 8 (4) 295-318.

Nelson, K. (1971). Accommodation of visual tracking patterns in human infants to object movement patterns. J. Experimental Child Psychology, 12(2), 182-96.

Pfeifer, R. and Iida, F. (2003). Embodied Artificial Intelligence: Trends and Challenges. LNCS, volume 3139, 1-26. Springer.

Pfeifer, R. and Scheier, C. (1999). Understanding Intelligence. MIT Press.

Piaget, J. (1952). The Origins of Intelligence in Children. International Univ. Press.

Schmid, U. and Raedt, L. D. (2019). Approaches and Applications of Inductive Programming. URL

=<https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=19202>

Searle, J. R. (1980). Minds, Brains and Programs. Behavioral and Brain Sciences, 3, 417–457.

Summers, P. D. (1977). A methodology for LISP program construction from examples. J. ACM, 24(1), 161–175.

Szymanik, J. (2004). Computational Semantics for Monadic Quantifiers in Natural Language, MA Thesis, Warsaw University.

Taddeo, M. and Floridi, L. (2005). Solving the Symbol Grounding Problem: A Critical Review of Fifteen Years of Research. J. Experimental & Theoretical Artificial Intelligence, 17, 419-445.

Tichy, P. (1969). Intension in terms of Turing Machines. Studia Logica, XXIV, 7–23.

Varela, F. J., Maturana, H. R. and Uribe, R. (1974). Autopoiesis: the organization of living systems: its characterization and a model. BioSystems 5, 187-196.

Ziemke, T. (2016). The body of knowledge: On the role of the living body in grounding embodied cognition. BioSystems 148, 4-11.