A Robotic Program that Acquires Concepts and Begins Introspection

Kenzo Iwama

Abstract
This paper firstly describes concept acquisition by a robot, and then explains a robotic program that acquires concepts. Concept acquisition by a robot can be described as follows. Suppose one has given the robot instances of a concept one after another. Then the one gives the robot a beginning part of an instance new to the robot, and sees how the robot behaves. If the robot finishes the rest of the instance, one can say that the robot has acquired the concept. The paper explains our robotic program that shows the behavior just described. The program has a preprogrammed program and components. The preprogrammed program generates programs by assembling components in such a way that the assemblies match inputs to the robot and robotic actions guided by a person. The preprogrammed program generalizes assemblies of components and applies generalized assemblies to a new instance. As a result of the application, the robot, given a beginning part of the instance, finishes the rest of the instance.

Key Words: robotic program, concepts, introspection, artificial mind

1. Introduction
This paper is concerned with acquisition of concepts and introspection by a machine. Acquisition of concepts by the machine has been a major theme from the beginning of computer science and robotics. Turing test devised by Turing (e.g., 1950) gives us a guideline to judge if the computer can be as intelligent as a human. The guideline tells that one gives questions to both a computer and a human, and that the one gets two sets of answers, one set from the computer and the other set from the human, without knowing which has made which set of answers. If the one cannot tell which set the computer made after the one examine the two sets, the computer is as intelligent as the human. AI researchers have investigated what theories and techniques can make computers intelligent and/or acquire concepts. McDermott (2001), McCarthy (2002), and others summarize the state-of-the-art of AI works.

However, Chinese room thought experiment by Searle (1980) demonstrates that a current AI program does not acquire concepts. The experiment is the
following: Suppose that a person who does not know any Chinese stays in a closed room. And suppose the person simulates the AI program. Namely, the person, through its window of the closed room, receives two kinds of inputs; one kind is strings written in Chinese and the other kind is mechanical steps for generating outputs from the inputs written in Chinese. The person appears to understand concepts conveyed in the input strings when the person correctly generates outputs from the inputs. But the person inside the room just follows the steps, and does not understand meanings of Chinese words. AI research has not yet shown mechanical steps by which a person understands meanings of Chinese words.

Much research needs to be conducted to develop a machine that acquires concepts; in particular, it seems necessary to study concepts acquired by very young children and to show mechanical steps that acquire such concepts. The steps describe the concepts within the steps and to apply them in a new situation.

This paper describes how a preprogrammed robotic program generates new programs by assembling components, and explains that generated programs represent concepts within the programs. The next section describes acquisition of concepts by a robot and argues that execution of assembled components makes the robot begin introspection.

Instances given to a robot must have generated links to invoke actions.

| words: “how many gray circles” |
| action: counting gray circle |
| results: has counted 2 gray circles |
| action: counting gray circle |
| words: “3” |

Objects in the robotic view

Fig. 1. Counting gray circles, links between words and actions, and links between action results and the next actions. Suppose instances given to a robot include words “how many” and actions of counting objects. Suppose the robot has not received “how many gray circles”, but it becomes able to count gray circles. Then the robot must have mechanism to link words “how many gray circles” and actions to count gray circles. The robot must also have mechanism to describe which the robot has counted (namely, action results) and to decide which the robot is going to count next.

2. Acquisition of Concept and Introspection

2.1 Concept Acquisition by a Robot

Suppose that a robot has not a concept, say \( C_p \), and a person has given a robot instances, from the beginning till the end, of the concept, \( C_p \). Suppose further the person has not given a particular instance, say \( I_p \), to the robot. Now the person gives the robot a beginning part of the instance, \( I_p \), and sees how the robot behaves. If the robot finishes the rest, one can say the robot has acquired the concept, \( C_p \). For example, a person would give the robot instances of counting various objects as follows; put 4 apples in the robotic view, and the person has given words “how many apples”. The person has guided the robot to move the apples one by one while the person has given the robot words “1”, “2”, and up to the number of the apples, “4”. Then the person has given words “4 apples”. Suppose the person has

\[ C_p \text{ of apples: } \{ \text{apple1, apple2, ..., apple4} \} \]
already given words “pick gray circles” while the person has guided the robot to pick gray circles, but has not guided the robot to count the gray circles. Then one can say the robot has acquired the concept of counting objects\(^3\) if the robot outputs the number of the gray circles in front of the robot when the person gives the robot words "how many gray circles”.

### 2.2 Robotic Introspection

This subsection argues that a robot must do introspection when the robot, given a beginning part of an instance, finishes the rest of the instance. One can assume that without loss of generality instances of a concept have inputs of symbolic words and a sequence of actions. Counting the number of objects is such an example. Suppose a person has never given the robot a particular instance that consists of symbolic words, say \(w_d\), and actions, say \(a_c\). But the robot becomes able to conduct the actions, \(a_c\), after the robot has received the words, \(w_d\). Then the robot must have at least the following: 1) a link between the words \(w_d\) and the actions \(a_c\), and 2) a link\(^4\) between action results and the next actions. Figure 1 illustrates the links. Note that the person has not programmed the links or has not input the links explicitly. The robot must have mechanism, say \(m_a\), to generate the links between the specific words, \(w_d\), and the specific actions, \(a_c\). The meanings behind the links must be the same as those behind many instances of the concept. Therefore the mechanism, \(m_a\), generates the links between specific words and actions in such a way that the links carry the meanings behind the many instances. One can say that the mechanism has a level to deal with specific words and specific actions, and another level to deal with the meanings behind the instances; in other words, the robot begins introspection.

### 3. Explanation of a Robotic Program

Our preprogrammed robotic program generates programs by assembling components and generalizes the programs; each generated program represents a robotic action in a specific and/or a general setting. A key feature of the program is that the preprogrammed program tries to achieve consistency among inputs and generated programs.

Iwama (2005) and Iwama (2006) describe sample programs that are tested in simulated worlds. The sample programs receive symbolic words, visual inputs such as distances from a robot, and actions guided by a human. The symbolic words are given in codes instead of visual and/or auditory sensor values.

#### 3.1 Outline of a Program

Our program has a preprogrammed program and components. The preprogrammed program roughly conducts the following: (1) It forms assemblies of components and sequences of assemblies as the program gets inputs\(^5\), and keeps the assemblies in its memory. In the case of counting the number of objects, the preprogrammed program forms a sequence to count particular objects. Figure 2 and 3 illustrate an example of a sequence of assembled components. (2) It generalizes sequences of assembled components, detects relations among the components, and stores them in its memory. Figure 4 shows an example of a generalized sequence. (3) The program retrieves generalized sequences of assembled components from its memory to make them match new inputs (a new instance) by merging the generalized sequences of assembled components. Here the merged components satisfy the relations detected at step (2). Figure 5 describes an overview of the program, and Figure 6 illustrates major steps of the program.

---

\(^3\) It can be argued that the number of objects cannot be found without actions of counting the objects. Although Figure 1 suggests it, this paper does not argue it.

\(^4\) A program could have all the possible preprogrammed links among the next actions, word inputs, and action results. By selecting a preprogrammed link, it would take the next action after getting word inputs or results of the previous action. But it is practically impossible that the robot has all the possible preprogrammed links by using currently available computing technology.

\(^5\) Inputs to a robot are continuous in time. But this paper does not discuss how to segment the continuous inputs into discrete ones.
Figure 2. A sample sequence of assembled components. Inputs to a robotic program are symbolic words such as “get 3 apples”, observation features such as distances, gray levels, and motions. At each step, an assembly is formed, and a sequence is formed as the inputs are given to the robot. A row of three boxes in the figure illustrates an assembly of components; one box includes symbolic words, the second box describes observation features, and the third describes robotic actions. Relations “same” are found among symbols, observation features, and motions.

Figure 3. A sample sequence of assembled components. Suppose a robot is given words “what is this” in front of oranges (in1 in the figure). The program retrieves, from its memory, an assembly that includes words “oranges” and observation features of oranges (ret in the figure), and other assemblies (omitted in the figure). And suppose that the robot is given words “oranges” (in2 in the figure). Then the sequence described as in the figure is formed, and relations “same” are detected among the symbolic words and observation features.

Figure 4. A sample of a generalized sequence of assembled components. After a program receives many inputs described as in Figure 3, the program generalizes them. Symbolic words and observation features that are common remain in the many assemblies, and relations common in the many assemblies remain there too. But different ones are replaced by tokens.
Components describe observation features, robotic actions, and symbolic words given by a human. The observation features include features of objects such as colors, texture, distances from a robot, and also they include virtual features like ‘group’ of two or three objects in a small area. The motions include those to do picking, grasping, putting (or releasing), touching, going forward, making right, making left, stopping, jumping, and other movements. The preprogrammed program uses measures to describe matching degrees among inputs and components, and to find similarity (or sameness) among components.

3.2 Generalization of Assembled Components
When our program has stored many sequences of assembled components in its memory, the program generalizes them and stores the generalized sequences of assembled components. Our sample program uses a very simple generalization method. The program first picks up sequences of assembled components that have something in common; for example, the program picks up assembled components that have words “what is this” as in Figure 3. Then the program replaces different observation features, motions, and words by tokens to memorize that some features, motions, and words occur in the components. At the same time, the program finds relationships “same” that hold among components in sequences of assemblies. For example, Figure 4 illustrates that words describing specific objects are replaced by tokens, and observation features specific to the objects are replaced by tokens.

3.3 Merging Sequences of Assembled Components
Our program retrieves assembled components from its memory that match new inputs when the program gets the inputs. The assembled components retrieved are usually more than one, and those are merged together in such a way that relationships found in the step of generalization are satisfied among the components. Then the merged components specify the next action the robot is supposed to be guided. Our sample program fills contents stored in sequences of assembled components into generalized sequences of assembled components so that the generalized sequences match current inputs. Figure 7 illustrates an example of merging assembled components when the robot is told to pick 3 pebbles. The program retrieves, from its memory, a generalized sequence of making actions 3 times, a sequence of picking something, and a sequence of describing a pebble. These sequences are put together to form new sequences that describe the next possible inputs as well as actions possibly guided by a person.

3.4 Consistency and Outputs
Inputs to our robotic program do not tell which inputs become outputs of the robot. But the robot is going to make outputs as a result of trying to keep consistency among inputs and assembled components retrieved from its memory.

When inputs are given to the program, it tries to reach consistency among the inputs and assembled components, say cac, retrieved from the memory. Suppose assembled components, say nac, follow the assembled components, cac. If the robot does not receive the next inputs or is not guided to do for a while, the robot makes outputs as described in the assembled components, nac. The outputs change an outside world, and new inputs from the changed world become consistent with the assembled components, nac (Figure 5).
Iwama, K. A robotic program, concepts, and introspection

**Figure 5.** An overview of a robotic program. Given inputs, the program retrieves assemblies of components from its memory (labeled Ret in the Figure) that match the inputs. The program merges them (labeled Mg in the Figure). Merged assemblies, say nac, follow the merged assemblies, say cac, and the merged assemblies, nac, become outputs of the robotic program. The outputs change a world that would change the inputs to the robot, and the changed inputs would match the merged assemblies, nac. The outputs would further change the world, and the inputs from the changed world would cause the robotic program to retrieve assemblies of components (that may follow nac) from the memory.

**Figure 6.** Major steps of a program. Retrieve assemblies (a1, and a2 in the figure) from the memory if they match inputs (m1 and m2 in the figure). Merge them (mg in the figure) in a way that relations "same" among the components hold. Make a merged assembly, in2, an output (out in the figure), so that the output changes the outside world and the inputs from the changed world match components in the output (mGoal in the figure).
Iwama, K. A robotic program, concepts, and introspection

4. Discussion

4.1 Acquisition of Concept and Chinese Room Thought Experiment

Suppose a person simulates our robotic program in a closed room. The person is guided to move its hands and / or legs while the person receives visual inputs as well as Chinese words. The person makes relations between the Chinese words and the actions on some objects. Later, if the person receives Chinese words, the person uses the relations between the Chinese words and actions, and becomes able to execute the related actions. Then one can say the person understands meanings of Chinese words.

Figure 7. An example of merging sequences of assembled components. Suppose that a robot is told to pick 3 pebbles. The robotic program retrieves, from its memory, assembled components that include “3”, those that include “pebble”, and those that include “pick”. When a robot has picked 2 pebbles, the robotic program has retrieved assemblies, merged them together, and used them (out in the figure). Assembled components with symbolic words “3” follow assembled components that have “2” in their sequence. By using relations “same”, the program also retrieves assemblies that include pebbles and assemblies that includes action, pick (ext in the figure). The program uses relations “same” to merge the assemblies together, and forms an output to pick another pebble (mgd and out in the figure).
4.2 Issues

Much needs to be investigated to make a robot acquire complex concepts. One may need to take two directions to do so. One direction is to let the robot conduct complex actions such as measuring the length of an object. It seems crucial to develop programs that describe movements of its focal point in their programs and let the robot apply the movements of the focal point for conducting actions; an example is to let the robot place its focal point on a short bar of a measure, move to the next short bar, and finally get the length of an object.

The other direction may be to study robotic goals; it seems that a robotic goal plays a role to form sequences in a way that one sequence includes one type of actions and another sequence includes another type of actions and the both sequences reach the goal. The robotic goal would be to do actions as instructed by a person. The goal drives the robotic program into forming alternative sequences of actions. For example, the robot has been given words “take a ball” by a person, and the robot has moved its hand at a speed already learned. But the ball has happened to be rolling fast, and the ball has gone. Now, without the goal how does the robot see if it has failed to take the ball? How does the robot form an alternative sequence of actions such as moving its hand quickly to take a ball rolling fast?

It seems that achieving consistency is a key to make the robotic program deal with its goal. Symbolic words such as “take a ball” can be linked to a sequence of actions and its results of the actions. The robot, given the symbolic words, retrieves the results of the actions from its memory. The results of the actions would let the robot see if it has made a success (namely, current results become consistent with the results of the actions retrieved from the memory), and mechanism of detecting inconsistency gives the robot a trigger to form alternative sequences of actions and memorize them for later use.

References


Accessed date: 1/Dec/2005


Turing A Computing Machinery and Intelligence. Mind 1950;59:236