International Journal of Research (IJR)

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

CRITICAL REVIEW OF THE FIFTH GENERATION OF PROGRAMMING LANGUAGE: THE PAST, PRESENT AND THE FUTURE.

OKEMIRI HENRY ANAYO
Department of Mathematice, Statistics, Computer Science and Informatics.
Federal University Ndufu Alike Ikwo, Ebonyi State.
E-mail: henry.okemiri@funai.edu.ng

ABSTRACT

Programming languages are the tools that allow communication between the computer and the developer. Far from being a static tool, programming languages evolve – they are created, constantly change, and frequently disappear over the course of their use. This article discusses the needs and forces that have shaped the evolution of fifth programming languages, their past, present and the future.

Keywords: Fifth Generation, Programming language, Artificial Intelligence

Introduction

A fifth generation (programming) language (5GL) is a grouping of programming languages build on the premise that a problem can be solved, and an application built to solve it, by providing constraints to the program (constraint-based programming), rather than specifying algorithmically how the problem is to be solved (imperative programming). (Hook, Audrey A, 2000).
In essence, the programming language is used to denote the properties, or logic, of a solution, rather than how it is reached. Most constraint-based and logic programming languages are 5GLs. A common misconception about 5GLs pertains to the practice of some 4GL vendors to denote their products as 5GLs, when in essence the products are evolved and enhanced 4GL tools.

These types of languages were also built upon Lisp, many originating on the Lisp machine, such as ICAD. Then, there are many frame languages, such as KL-ONE.

Dijkstra, Edsger W (1980), fifth-generation languages were considered to be the way of the future, and some predicted that they would replace all other languages for system development, with the exception of low-level languages. Most notably,

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

from 1982 to 1993 Japan put much research and money into their fifth generation computer systems project, hoping to design a massive computer network of machines using these tools.

The ability to perform many complex tasks at one time was expanded and revolutionized with the introduction of the microprocessor in the early 70's. Now what took up a whole room could rest gently on a fingertip. Microprocessors were the beginning for a fury of technological advancements that includes computerized cars, appliances and smart phones. Everything has become smarter, faster and smaller. They have also become integrated. With the advent of the microprocessor came the ability to link computers together in a network. The birth of the Internet and all of its wonders are attributed to the birth of the microchip.

All of this has led to the ability to program computers to imitate the ability to think. While no computer or device can truly think on its own (sorry HAL) it is able to simulate many decision making functions that have helped to improve the lives and fun of humans.

Artificial intelligence is the realm of programming where devices are enabled with the ability to think and react to the environment around it. The fields of gaming, robotics, voice recognition, and real life simulation all center on perfecting the science of artificial intelligence Back in the earliest stages of computing, computers contained vacuum tubes and magnetic drums. They were large, expensive and could only perform one task at a time. They were also prone to malfunctions and had the self-destructive inclination to overheat due to the vast amount of electricity it used and heat it generated.

However, as larger programs were built, the flaws of the approach became more apparent. It turns out that, given a set of constraints defining a particular problem, deriving an efficient algorithm to solve it is a very difficult problem in itself. This crucial step cannot yet be automated and still requires the insight of a human programmer.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

AI programmers started with basic algorithms for reasoning and progressed to using probability and economic theories to help create the ability to solve more complex issues. Advances in the field of fuzzy logic have introduced the ability to solve problems where there is no clear answer of right or wrong. While more complex problem solving is possible, the amount of computing power necessary expands exponentially when the solution is difficult. Research is ongoing to find ways to make problem solving more efficient and less cost prohibitive.

In the military, AI has helped enhance simulators so that training soldiers for unexpected problems that arise in different peacekeeping missions around the world. Artificial Intelligence is also prevalent in many of the world's spy networks to help determine the probability of certain actions occurring in highly volatile parts of the world.

In the automobile industry, robots are used in manufacturing, but AI has crept into the vehicles themselves. From global positioning systems to warning a driver of a potential hazard in front of them, artificial intelligence is present. Some cars can even apply the break or swerve the vehicle out of harm's way if necessary.

Artificial intelligence has also assisted in the development of voice recognition software (VRS). While unable to understand the words, VAR has allowed people who have disabilities to speak into a microphone and see the words appear on the screen. Although people need to speak slowly and clearly in order to work properly, and it is not 100% accurate, VAR is also now on many hand held devices. Being able to surf the web or send a text has become easier than ever, and more advancements are on the way.

This can be seen in the robot that won on jeopardy and the customer service avatars now seen on websites and as office assistants. One virtual office assistant (HAL's ancestor perhaps) can hold conversation with the user. As well as remind him to pick up a gallon of milk on the way home.

A programming language allows a developer to translate logical real-world actions into operations that can be performed on computer hardware. In effect, it is a way

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

to translate concrete real-world desires into computer-world operations. Programming languages advance by extending the number of operations programmers can perform without thinking about them – thus making it easier to say the things they want to say. In effect, these advances hide the complexity of what is going on underneath the hood and raise the level of abstraction that programmers think about when they program. If a programmer wants to say something to the computer, and he/she finds that the current language has difficulty in saying it, then he/she develops a new language or extends an existing language. Advances in programming languages tend to increase the intellectual distance between program statements and what the computer hardware actually does. The language then does more of our work, while decreasing the distance between the programs written and the real world, allowing us to solve real-world problems in the context and language of the real world. On a subtler note, programming languages, software,
computer scientists, etc. exert an influence on the real world also, drawing it ever closer to the software realm. Inevitably, a new programming language enables a programmer to express an idea or concept in a simpler, more readable manner than what had come before. This simpler, more readable manner allows us to create code that is easier to verify, easier to code, and easier to debug. In essence, the more powerful a programming language is, the easier it is to express complex ideas in a simple manner

A **fifth generation programming language** (abbreviated as **5GL**) is a programming language based on solving using constraints given to the program, rather than using an algorithm written by a programmer. Most constraint-based and logic programming languages and some declarative languages are fifth-generation languages. While fourth-generation programming languages are designed to build specific programs, fifth-generation languages are designed to make the computer solve a given problem without the programmer. This way, the programmer only needs to worry about what problems need to be solved and what conditions need to be met, without worrying about how to implement a routine or algorithm to solve them. Fifth-generation languages are used mainly in artificial intelligence research. Prolog, OPS5, and Mercury are examples of fifth-generation languages. **Artificial intelligence programming language,** a computer language

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

developed expressly for implementing artificial intelligence (AI) research. In the course of their work on the Logic Theorist and GPS, two early AI programs, Allen Newell and J. Clifford Shaw of the Rand Corporation and Herbert Simon of Carnegie Mellon University developed their Information Processing Language (IPL), a computer language tailored for AI programming. At the heart of IPL was a highly flexible data structure that they called a list. A list is simply an ordered sequence of items of data. Some or all of the items in a list may themselves be lists. This scheme leads to richly branching structures.

In 1960 John McCarthy, a computer scientist at the Massachusetts Institute of Technology(MIT), combined elements of IPL with the lambda calculus (a formal mathematical-logical system) to produce the programming language LISP (List Processor), which remains the principal language for AI work in the United States. (The lambda calculus itself was invented in 1936 by the Princeton University logician Alonzo Church while he was investigating the abstract *Entscheidungsproblem*, or "decision problem," for predicate calculus—the same problem that the British mathematician and logician Alan Turing had been attacking when he invented the universal Turing machine.)

The logic programming language PROLOG (Programmation en Logique) was conceived byAlain Colmerauer at the University of Aix-Marseille, France, where the language was first implemented in 1973. PROLOG was further developed by the logician Robert Kowalski, a member of the AI group at the University of Edinburgh. This language makes use of a powerful theorem-proving technique known as resolution, invented in 1963 at the U.S. Atomic Energy Commission's Argonne National Laboratory in Illinois by the British logician Alan Robinson. PROLOG can determine whether or not a given statement follows logically from other given statements. For example, given the statements "All logicians are rational" and "Robinson is a logician," a PROLOG program responds in the affirmative to the query "Robinson is rational?" PROLOG is widely used for AI work, especially in Europe and Japan.

Researchers at the Institute for New Generation Computer Technology in Tokyo have used PROLOG as the basis for sophisticated logic programming languages. Known as fifth-generation languages, these are in use on nonnumerical parallel computers developed at the Institute.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

Other recent work includes the development of languages for reasoning about time-dependent data such as "the account was paid yesterday." These languages are based on tense logic, which permits statements to be located in the flow of time. (Tense logic was invented in 1953 by the philosopher Arthur Prior at the University of Canterbury, Christchurch, New Zealand.)

Artificial intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—as, for example, discovering proofs for mathematical theorems or playing chess—with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge. On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, and voice or handwriting recognition.

What is intelligence?

All but the simplest human behaviour is ascribed to intelligence, while even the most complicated insect behaviour is never taken as an indication of intelligence. What is the difference? Consider the behaviour of the digger wasp, *Sphex ichneumoneus*. When the female wasp returns to her burrow with food, she first deposits it on the threshold, checks for intruders inside her burrow, and only then, if the coast is clear, carries her food inside. The real nature of the wasp's instinctual behaviour is revealed if the food is moved a few inches away from the entrance to her burrow while she is inside: on emerging, she will repeat the whole procedure as often as the food is displaced. Intelligence—conspicuously absent in the case of *Sphex*—must include the ability to adapt to new circumstances.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

Psychologists generally do not characterize human intelligence by just one trait but by the combination of many diverse abilities. Research in AI has focused chiefly on the following components of intelligence: learning, reasoning, problem solving, perception, and using language.

Learning

There are a number of different forms of learning as applied to artificial intelligence. The simplest is learning by trial and error. For example, a simple computer program for solving mate-in-one chess problems might try moves at random until mate is found. The program might then store the solution with the position so that the next time the computer encountered the same position it would recall the solution. This simple memorizing of individual items and procedures—known as rote learning—is relatively easy to implement on a computer. More challenging is the problem of implementing what is called generalization. Generalization involves applying past experience to analogous new situations. For example, a program that learns the past tense of regular English verbs by rote will not be able to produce the past tense of a word such as *jump* unless it previously had been presented with *jumped*, whereas a program that is able to generalize can learn the "add *ed*" rule and so form the past tense of*jump* based on experience with similar verbs.

Reasoning

To reason is to draw inferences appropriate to the situation. Inferences are classified as eitherdeductive or inductive. An example of the former is, "Fred must be in either the museum or the café. He is not in the café; therefore he is in the museum," and of the latter, "Previous accidents of this sort were caused by instrument failure; therefore this accident was caused by instrument failure." The most significant difference between these forms of reasoning is that in the deductive case the truth of the premises guarantees the truth of the conclusion, whereas in the inductive case the truth of the premise lends support to the conclusion without giving absolute assurance. Inductive reasoning is common in science, where data are collected and tentative models are developed to describe and predict future behaviour—until the appearance of anomalous data forces the model to be revised. Deductive reasoning is common in mathematics and logic,

where elaborate structures of irrefutable theorems are built up from a small set of basic axioms and rules.

There has been considerable success in programming computers to draw inferences, especially deductive inferences. However, true reasoning involves more than just drawing inferences; it involves drawing inferences *relevant* to the solution of the particular task or situation. This is one of the hardest problems confronting AI.

Problem solving

Problem solving, particularly in artificial intelligence, may be characterized as a systematic search through a range of possible actions in order to reach some predefined goal or solution. Problem-solving methods divide into special purpose and general purpose. A special-purpose method is tailor-made for a particular problem and often exploits very specific features of the situation in which the problem is embedded. In contrast, a general-purpose method is applicable to a wide variety of problems. One general-purpose technique used in AI is means-end analysis—a step-by-step, or incremental, reduction of the difference between the current state and the final goal. The program selects actions from a list of means—in the case of a simple robot this might consist of PICKUP, PUTDOWN, MOVEFORWARD, MOVEBACK, MOVELEFT, and MOVERIGHT—until the goal is reached. Many diverse problems have been solved by artificial intelligence programs. Some examples are finding the winning move (or sequence of moves) in a board game, devising mathematical proofs, and manipulating “virtual objects” in a computer-generated world.

Perception

In perception the environment is scanned by means of various sensory organs, real or artificial, and the scene is decomposed into separate objects in various spatial relationships. Analysis is complicated by the fact that an object may appear different depending on the angle from which it is viewed, the direction and intensity of illumination in the scene, and how much the object contrasts with the surrounding field.

At present, artificial perception is sufficiently well advanced to enable optical sensors to identify individuals, autonomous vehicles to drive at moderate speeds on

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

the open road, and robots to roam through buildings collecting empty soda cans. One of the earliest systems to integrate perception and action was FREDDY, a stationary robot with a moving television eye and a pincer hand, constructed at the University of Edinburgh, Scotland, during the period 1966–73 under the direction of Donald Michie. FREDDY was able to recognize a variety of objects and could be instructed to assemble simple artifacts, such as a toy car, from a random heap of components.

Language

A language is a system of signs having meaning by convention. In this sense, language need not be confined to the spoken word. Traffic signs, for example, form a minilanguage, it being a matter of convention that [hazard] means "hazard ahead" in some countries. It is distinctive of languages that linguistic units possess meaning by convention, and linguistic meaning is very different from what is called natural meaning, exemplified in statements such as "Those clouds mean rain" and "The fall in pressure means the valve is malfunctioning."

An important characteristic of full-fledged human languages—in contrast to birdcalls and traffic signs—is their productivity. A productive language can formulate an unlimited variety of sentences.

It is relatively easy to write computer programs that seem able, in severely restricted contexts, to respond fluently in a human language to questions and statements. (For examples see BTW: Dialogue with Parry and BTW: Dialogue with SHRDLU). Although none of these programs actually understands language, they may, in principle, reach the point where their command of a language is indistinguishable from that of a normal human. What, then, is involved in genuine understanding, if even a computer that uses language like a native human speaker is not acknowledged to understand? There is no universally agreed upon answer to this difficult question. According to one theory, whether or not one understands depends not only on one's behaviour but also on one's history: in order to be said to understand, one must have learned the language and have been trained to take one's place in the linguistic community by means of interaction with other language users.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

Methods and goals in AI

Symbolic vs. connectionist approaches

AI research follows two distinct, and to some extent competing, methods, the symbolic (or "top-down") approach, and the connectionist (or "bottom-up") approach. The top-down approachseeks to replicate intelligence by analyzing cognition independent of the biological structure of the brain, in terms of the processing of symbols—whence the *symbolic* label. The bottom-up approach, on the other hand, involves creating artificial neural networks in imitation of the brain's structure—whence the *connectionist* label.

To illustrate the difference between these approaches, consider the task of building a system, equipped with an optical scanner, that recognizes the letters of the alphabet. A bottom-up approach typically involves training an artificial neural network by presenting letters to it one by one, gradually improving performance by "tuning" the network. (Tuning adjusts the responsiveness of different neural pathways to different stimuli.) In contrast, a top-down approach typically involves writing a computer program that compares each letter with geometric descriptions. Simply put, neural activities are the basis of the bottom-up approach, while symbolic descriptions are the basis of the top-down approach.

In *The Fundamentals of Learning* (1932), Edward Thorndike, a psychologist at Columbia University, New York City, first suggested that human learning consists of some unknown property of connections between neurons in the brain. In *The Organization of Behavior* (1949),Donald Hebb, a psychologist at McGill University, Montreal, Canada, suggested that learning specifically involves strengthening certain patterns of neural activity by increasing the probability (weight) of induced neuron firing between the associated connections. The notion of weighted connections is described in a later section, Connectionism.

In 1957 two vigorous advocates of symbolic AI—Allen Newell, a researcher at the RAND Corporation, Santa Monica, California, and Herbert Simon, a psychologist and computer scientist at Carnegie Mellon University, Pittsburgh, Pennsylvania—summed up the top-down approach in what they called the physical symbol system hypothesis. This hypothesis states that processing structures of symbols is sufficient, in principle, to produce artificial intelligence in a digital computer and

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

that, moreover, human intelligence is the result of the same type of symbolic manipulations.

During the 1950s and ’60s the top-down and bottom-up approaches were pursued simultaneously, and both achieved noteworthy, if limited, results. During the 1970s, however, bottom-up AI was neglected, and it was not until the 1980s that this approach again became prominent. Nowadays both approaches are followed, and both are acknowledged as facing difficulties. Symbolic techniques work in simplified realms but typically break down when confronted with the real world; meanwhile, bottom-up researchers have been unable to replicate the nervous systems of even the simplest living things. *Caenorhabditis elegans*, a much-studied worm, has approximately 300 neurons whose pattern of interconnections is perfectly known. Yet connectionist models have failed to mimic even this worm. Evidently, the neurons of connectionist theory are gross oversimplifications of the real thing.

Strong AI, applied AI, and cognitive simulation

Employing the methods outlined above, AI research attempts to reach one of three goals: strong AI, applied AI, or cognitive simulation. Strong AI aims to build machines that think. (The term *strong AI* was introduced for this category of research in 1980 by the philosopher John Searle of the University of California at Berkeley.) The ultimate ambition of strong AI is to produce a machine whose overall intellectual ability is indistinguishable from that of a human being. As is described in the section Early milestones in AI, this goal generated great interest in the 1950s and ’60s, but such optimism has given way to an appreciation of the extreme difficulties involved. To date, progress has been meagre. Some critics doubt whether research will produce even a system with the overall intellectual ability of an ant in the forseeable future. Indeed, some researchers working in AI’s other two branches view strong AI as not worth pursuing.

Applied AI, also known as advanced information processing, aims to produce commercially viable “smart” systems—for example, “expert” medical diagnosis systems and stock-trading systems. Applied AI has enjoyed considerable success, as described in the section Expert systems.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

In cognitive simulation, computers are used to test theories about how the human mind works—for example, theories about how people recognize faces or recall memories. Cognitive simulation is already a powerful tool in both neuroscience and cognitive psychology.

Alan Turing and the beginning of AI

Theoretical work

The earliest substantial work in the field of artificial intelligence was done in the mid-20th century by the British logician and computer pioneer Alan Mathison Turing. In 1935 Turing described an abstract computing machine consisting of a limitless memory and a scanner that moves back and forth through the memory, symbol by symbol, reading what it finds and writing further symbols. The actions of the scanner are dictated by a program of instructions that also is stored in the memory in the form of symbols. This is Turing's stored-programconcept, and implicit in it is the possibility of the machine operating on, and so modifying or improving, its own program. Turing's conception is now known simply as the universal Turing machine. All modern computers are in essence universal Turing machines.

During World War II, Turing was a leading cryptanalyst at the Government Code and Cypher School in Bletchley Park, Buckinghamshire, England. Turing could not turn to the project of building a stored-program electronic computing machine until the cessation of hostilities in Europe in 1945. Nevertheless, during the war he gave considerable thought to the issue of machine intelligence. One of Turing's colleagues at Bletchley Park, Donald Michie (who later founded the Department of Machine Intelligence and Perception at the University of Edinburgh), later recalled that Turing often discussed how computers could learn from experience as well as solve new problems through the use of guiding principles—a process now known as heuristic problem solving.

Turing gave quite possibly the earliest public lecture (London, 1947) to mention computer intelligence, saying, "What we want is a machine that can learn from experience," and that the "possibility of letting the machine alter its own instructions provides the mechanism for this." In 1948 he introduced many of the central concepts of AI in a report entitled "Intelligent Machinery." However,

Turing did not publish this paper, and many of his ideas were later reinvented by others. For instance, one of Turing's original ideas was to train a network of artificial neurons to perform specific tasks, an approach described in the sectionConnectionism.

The Turing test

In 1950 Turing sidestepped the traditional debate concerning the definition of intelligence, introducing a practical test for computer intelligence that is now known simply as the Turing test. The Turing test involves three participants: a computer, a human interrogator, and a human foil. The interrogator attempts to determine, by asking questions of the other two participants, which is the computer. All communication is via keyboard and display screen. The interrogator may ask questions as penetrating and wide-ranging as he or she likes, and the computer is permitted to do everything possible to force a wrong identification. (For instance, the computer might answer, "No," in response to, "Are you a computer?" and might follow a request to multiply one large number by another with a long pause and an incorrect answer.) The foil must help the interrogator to make a correct identification. A number of different people play the roles of interrogator and foil, and, if a sufficient proportion of the interrogators are unable to distinguish the computer from the human being, then (according to proponents of Turing's test) the computer is considered an intelligent, thinking entity.

Early milestones in AI

The first AI programs

The earliest successful AI program was written in 1951 by Christopher Strachey, later director of the Programming Research Group at the University of Oxford. Strachey's checkers(draughts) program ran on the Ferranti Mark I computer at the University of Manchester, England. By the summer of 1952 this program could play a complete game of checkers at a reasonable speed.

Information about the earliest successful demonstration of machine learning was published in 1952. Shopper, written by Anthony Oettinger at the University of Cambridge, ran on the EDSACcomputer. Shopper's simulated world was a mall of eight shops. When instructed to purchase an item, Shopper would search for it, visiting shops at random until the item was found. While searching, Shopper would

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

memorize a few of the items stocked in each shop visited (just as a human shopper might). The next time Shopper was sent out for the same item, or for some other item that it had already located, it would go to the right shop straight away. This simple form of learning, as is pointed out in the introductory section What is intelligence?, is called rote learning.

The first AI program to run in the United States also was a checkers program, written in 1952 by Arthur Samuel for the prototype of the IBM 701. Samuel took over the essentials of Strachey's checkers program and over a period of years considerably extended it. In 1955 he added features that enabled the program to learn from experience. Samuel included mechanisms for both rote learning and generalization, enhancements that eventually led to his program's winning one game against a former Connecticut checkers champion in 1962.

Evolutionary computing

Samuel's checkers program was also notable for being one of the first efforts at evolutionary computing. (His program "evolved" by pitting a modified copy against the current best version of his program, with the winner becoming the new standard.) Evolutionary computing typically involves the use of some automatic method of generating and evaluating successive "generations" of a program, until a highly proficient solution evolves.

Today's leading proponent of evolutionary computing, John Holland, also wrote test softwarefor the prototype of the IBM 701 computer. In particular, he helped design a neural-network "virtual" rat that could be trained to navigate through a maze. This work convinced Holland of the efficacy of the bottom-up approach. While continuing to consult for IBM, Holland moved to the University of Michigan in 1952 to pursue a doctorate in mathematics. He soon switched, however, to a new interdisciplinary program in computers and information processing (later known as communications science) created by Arthur Burks, one of the builders of ENIAC and its successor EDVAC. In his 1959 dissertation, for most likely the world's first computer science Ph.D., Holland proposed a new type of computer—a multiprocessor computer—that would assign each artificial neuron in a network to a separate processor. (In 1985 Daniel Hillis solved the engineering

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

difficulties to build the first such computer, the 65,536-processor Thinking Machines Corporation supercomputer.)

Holland joined the faculty at Michigan after graduation and over the next four decades directed much of the research into methods of automating evolutionary computing, a process now known by the term *genetic algorithms*. Systems implemented in Holland's laboratory included a chess program, models of single-cell biological organisms, and a classifier system for controlling a simulated gas-pipeline network. Genetic algorithms are no longer restricted to "academic" demonstrations, however; in one important practical application, a genetic algorithm cooperates with a witness to a crime in order to generate a portrait of the criminal.

Logical reasoning and problem solving

The ability to reason logically is an important aspect of intelligence and has always been a major focus of AI research. An important landmark in this area was a theorem-proving program written in 1955–56 by Allen Newell and J. Clifford Shaw of the RAND Corporation and Herbert Simon of the Carnegie Mellon University. The Logic Theorist, as the program became known, was designed to prove theorems from *Principia Mathematica* (1910–13), a three-volume work by the British philosopher-mathematicians Alfred North Whitehead and Bertrand Russell. In one instance, a proof devised by the program was more elegant than the proof given in the books.

Newell, Simon, and Shaw went on to write a more powerful program, the General Problem Solver, or GPS. The first version of GPS ran in 1957, and work continued on the project for about a decade. GPS could solve an impressive variety of puzzles using a trial and error approach. However, one criticism of GPS, and similar programs that lack any learning capability, is that the program's intelligence is entirely secondhand, coming from whatever information the programmer explicitly includes.

English dialogue

Two of the best-known early AI programs, Eliza and Parry, gave an eerie semblance of intelligent conversation. (Details of both were first published in 1966.) Eliza, written by Joseph Weizenbaum of MIT's AI Laboratory, simulated a human therapist. Parry, written by Stanford University psychiatrist Kenneth Colby,

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

simulated a human paranoiac. Psychiatrists who were asked to decide whether they were communicating with Parry or a human paranoiac were often unable to tell. (For a sample conversation, see BTW: Dialogue with Parry.) Nevertheless, neither Parry nor Eliza could reasonably be described as intelligent. Parry's contributions to the conversation were canned—constructed in advance by the programmer and stored away in the computer's memory. Eliza, too, relied on canned sentences and simple programming tricks.

AI programming languages

In the course of their work on the Logic Theorist and GPS, Newell, Simon, and Shaw developed their Information Processing Language (IPL), a computer language tailored for AI programming. At the heart of IPL was a highly flexible data structure that they called a list. A list is simply an ordered sequence of items of data. Some or all of the items in a list may themselves be lists. This scheme leads to richly branching structures.

In 1960 John McCarthy combined elements of IPL with the lambda calculus (a formal mathematical-logical system) to produce the programming language LISP (List Processor), which remains the principal language for AI work in the United States. (The lambda calculus itself was invented in 1936 by the Princeton logician Alonzo Church while he was investigating the abstract *Entscheidungsproblem*, or "decision problem," for predicate logic—the same problem that Turing had been attacking when he invented the universal Turing machine.)

The logic programming language PROLOG (Programmation en Logique) was conceived by Alain Colmerauer at the University of Aix-Marseille, France, where the language was first implemented in 1973. PROLOG was further developed by the logician Robert Kowalski, a member of the AI group at the University of Edinburgh. This language makes use of a powerful theorem-proving technique known as resolution, invented in 1963 at the U.S. Atomic Energy Commission's Argonne National Laboratory in Illinois by the British logician Alan Robinson. PROLOG can determine whether or not a given statement follows logically from other given statements. For example, given the statements "All logicians are rational" and "Robinson is a logician," a PROLOG program responds in the

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

affirmative to the query "Robinson is rational?" PROLOG is widely used for AI work, especially in Europe and Japan.

Researchers at the Institute for New Generation Computer Technology in Tokyo have used PROLOG as the basis for sophisticated logic programming languages. Known as fifth-generation languages, these are in use on nonnumerical parallel computers developed at the Institute.

Other recent work includes the development of languages for reasoning about time-dependent data such as "the account was paid yesterday." These languages are based ontense logic, which permits statements to be located in the flow of time. (Tense logic was invented in 1953 by the philosopher Arthur Prior at the University of Canterbury, Christchurch, New Zealand.)

Microworld programs

To cope with the bewildering complexity of the real world, scientists often ignore less relevant details; for instance, physicists often ignore friction and elasticity in their models. In 1970Marvin Minsky and Seymour Papert of the MIT AI Laboratory proposed that likewise AI research should focus on developing programs capable of intelligent behaviour in simpler artificial environments known as microworlds. Much research has focused on the so-calledblocks world, which consists of coloured blocks of various shapes and sizes arrayed on a flat surface.

An early success of the microworld approach was SHRDLU, written by Terry Winograd of MIT. (Details of the program were published in 1972.) SHRDLU controlled a robot arm that operated above a flat surface strewn with play blocks. Both the arm and the blocks were virtual. SHRDLU would respond to commands typed in natural English, such as "Will you please stack up both of the red blocks and either a green cube or a pyramid." The program could also answer questions about its own actions. (For a sample, see BTW: Dialogue with SHRDLU.)Although SHRDLU was initially hailed as a major breakthrough, Winograd soon announced that the program was, in fact, a dead end. The techniques pioneered in the program proved unsuitable for application in wider, more interesting worlds. Moreover, the appearance that SHRDLU gave of understanding the blocks microworld, and English statements concerning it, was in fact an illusion. SHRDLU had no idea what a green block was.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 8
August 2021

References

1. Hook, Audrey A., et al. "A Survey of Computer Programming Languages Currently Used in the Department of Defense: An Executive Summary." CrossTalk 8.10 (Oct. 1995 www.stsc.hill.af.mil/crosstalk/1995/10/index.html>.
2. Flon, Lawrence. "On Research in Structured Programming." SIGPLAN Notices 10:10 (Oct. 1975).
3. Dijkstra, Edsger W. "Go To Statement Considered Harmful." Communications of the ACM 11.3 (Mar. 1968):147-148.
4. Webopedia. Online dictionary and search engine for computer and Internet technology <www.webopedia.com>.
5. Dijkstra, Edsger W. Selected Writings on Computing: A Personal Perspective. Springer-Verlag, 1982.
6. Edward Thorndike "*The Fundamentals of Learning"* (1932),