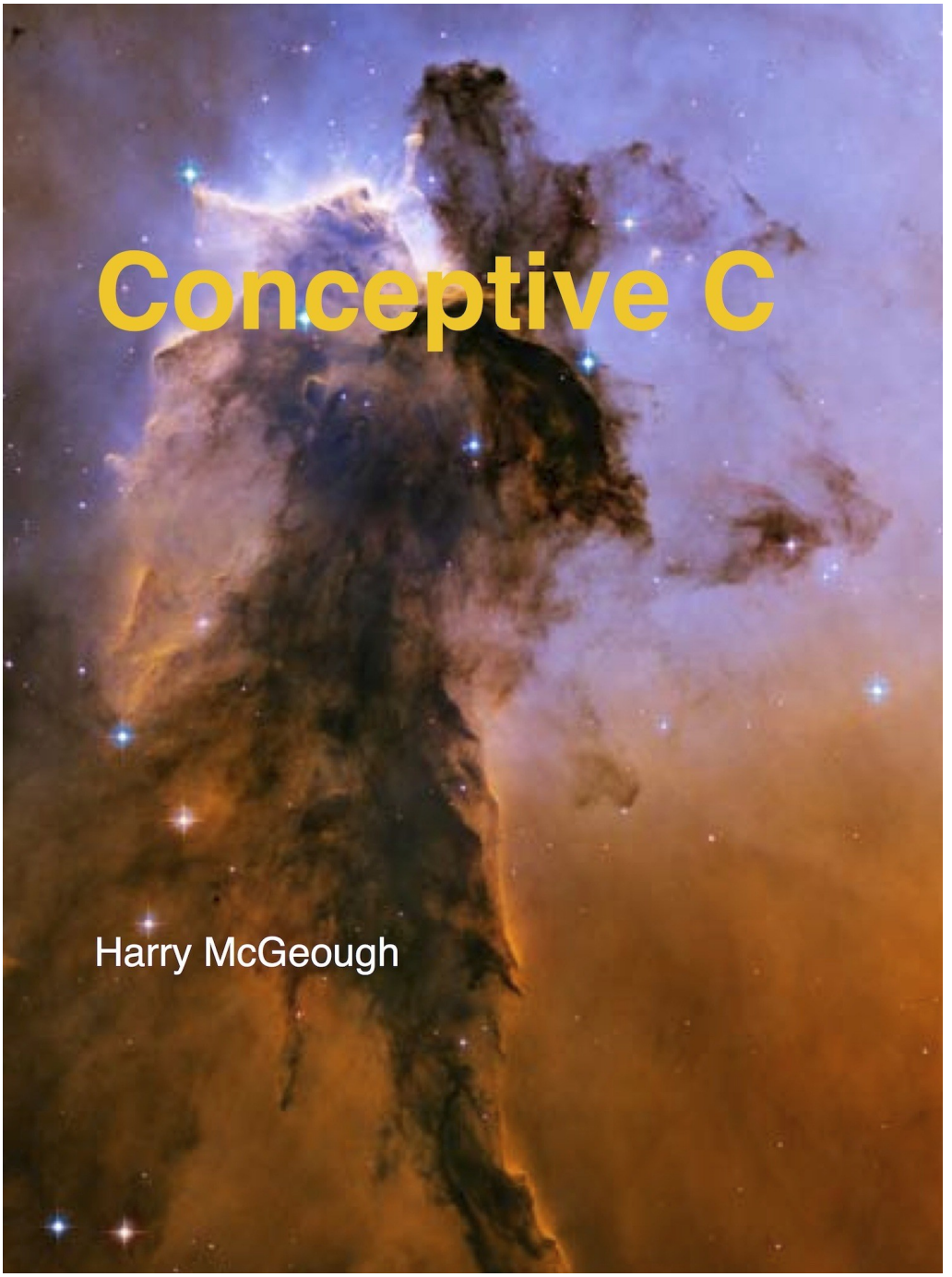
Conceptive C
Harry McGeough

Conceptive C

Harry McGeough

Version 1.0 Published by Harry McGeough at Smashwords

ISBN-13: 978-1465934888

ISBN-10: 1465934888

About Cover Image

A billowing tower of gas and dust rises from the stellar nursery known as the Eagle Nebula. This small piece of the Eagle Nebula is 57 trillion miles long (91.7 trillion km).

Credit: NASA, ESA, and The Hubble Heritage Team (STScI/AURA)

Contents

Instantiation
Protocols
Dynamic Typing
Forwarding
Categories
#import
Objective-C 2.0
Garbage Collection
Properties
Non-fragile Instance Variables
Fast Enumeration
Library Use
Analysis of the Language

Lisp

Connection to Artificial Intelligence
Symbolic expressions
Lists
Operators
Lambda Expressions
Atoms
Conses and Lists
S-expressions Represent Lists
List-Processing Procedures
Shared Structure
Self-evaluating Forms and Quoting
List Structure of Program Code
Evaluation and the Read–Eval–Print Loop
Object Systems

Conceptive C

Syntax
Messages
Interfaces and Implementations
Interface
Instantiation
Dynamic Typing
Changes from Objective-C
Machine Intelligence
New Ideas

Idea

Innate and Adventitious Ideas
Plato
René Descartes
John Locke
David Hume
Immanuel Kant

Rudolf Steiner
Wilhelm Wundt
Charles Sanders Peirce
G.F. Stout and J.M. Baldwin
Anthropology and the Social Sciences
Dr. Samuel Johnson
Validity of Ideas

Concept

What is a Concept?
Origin and Acquisition of Concepts
Pure Concepts
Conceptual Structure
One Possible Structure
The Dual Nature of Concepts
Conceptual Content
Concepts and Metaphilosophy
Concepts in Epistemology
Ontology of Concepts
Conceptual Empirical Investigations

Memory

Processes
Sensory Memory
Short-Term Memory
Long-Term Memory
Working Memory
Levels of Processing
Information Type
Temporal Direction
Physiology

Mind

Concept of Mind
Etymology
Mental Faculties
Mental Content
Cognitive Science
Philosophy of Mind
Mind / Body Perspectives
Psychology
Evolutionary Psychology
Evolution of the Human Mind
Animal intelligence
Artificial intelligence

Meme

Origins

Transmission
Memes as Discrete Units
Evolutions Influences on Memes

Cell

Anatomy
Prokaryotic Cells
Eukaryotic Cells
Genetic Material
Origin of the First Cell
Computer Cell

Soul

Etymology
Semantics
Philosophical views
Aristotle
Avicenna and Ibn Al-Nafis
Thomas Aquinas
Immanuel Kant
James Hillman
Philosophy of Mind
Buddhism
Judaism
Christianity
Roman Catholic Beliefs
Hinduism
Islam
Taoism
Zoroastrianism
Spirituality and New Age
Science

Understanding

Is Understanding Definable?
Understanding Basic English
Rules of word use
Operations - 100 words
400 General Words
Things - 200 Picture-able Words
Qualities - 100 Descriptive words
Qualities - 50 opposites

Consciousness

Etymology and Early History
In Philosophy
Is Consciousness a Valid Concept?
Is It a Single Thing?

How does it relate to the Physical World?
Why do people believe that other people are Conscious?
Are non-human animals are Conscious?
Could a Machine ever be Conscious?
Spiritual Approaches
Scientific Approaches
Biological Function and Evolution
States of Consciousness
Phenomenology
Assessment
The Stream of Consciousness

Thought Experiment

The Chinese Room
Chinese Room Thought Experiment
History
Computationalism
Strong AI
Strong AI as Functionalism
Computers vs. Machines vs. Brains
Intentionality vs. Consciousness
Strong AI vs. AI research
System Reply
Virtual Mind Reply
Finding the Meaning
Robot Reply
Derived Meaning
Commonsense Knowledge
Brain Simulator Reply
Formal Arguments

Mind Uploading

Simple Artificial Neural Network
Immortality/Backup
Speedup
Multiple / Parallel Existence
Computational Capacity
Simulation Model Scale
Scanning and Mapping Scale of an Individual
Serial Sectioning of a Brain
Brain Imaging
Neuroinformatics
Legal, Political and Economical Issues
Copying vs. Moving
Bekenstein Bound

Bots

Chatter Bot

ELIZA
IRC Bot
Avatar
Norman Spinrad
William Gibson
Neal Stephenson
Artificial Intelligence
Video Games
AIML
Categories
Patterns
Template
AIML 1.0 Tag Set

Preface

While writing this book I started quoting from Wikipedia. Then including sections, then whole parts of Wiki... I would highly recommend using http://www.wikipedia.com In fact there is quite a bit from Wiki in the book now. I do plan to edit some of it out.

This did start out as a simple question and answer Bot and grew. I now have an Apple 4S with Siri. So I have been exploring Wolfram Alpha's website.

I'm still trying to understand how I think to develop the language. It's a hunch, I have that it should be possible to write a program that understands things, if the right tools are available. I am hoping some of the stuff from Wiki will lead to improvements in Conceptive C Language.

I'm still not sure just yet whether Conceptual-C is in fact a conceptual Language. It is possible that in designing it, I may find that Objective-C already does what I'm trying to do. I'm still not at the stage of knowing what needs to go into the compiler yet.

What a month this has been first Steve Jobs (February 24, 1955 – October 5, 2011) of Apple died. My first computer was an Apple II, then I had a PC, then a Macintosh, a NEXT Station and my current computer, which I'm very happy with is an iMac. I saw Steve live at some of the Apple WWDC (developer conferences) and visited the Apple campus. I want to get an iPad and yes, I use an iPhone too.

Sad to that John McCarthy (September 4, 1927 – October 23, 2011) died just this week. He coined the term "artificial intelligence" (AI), invented the Lisp programming language and was highly influential in the early development of AI.

These things always seem to happen in three's now I've hear that Denis Ritchie (September 9, 1941 - October 12, 2011) is dead. He created C programming Language. He was the 'R' in K&R book "The C Programming Language".

I've included Lisp and AI in this book, because I feel it will help in implementing Conceptive-C. I have also included a sections on Memes, Mind Uploading, Neural Nets, The Chinese Room, and Bots, as interesting possible AI future ideas. They may get cut in the next version.

Articles used from Wikipedia

AI http://en.wikipedia.org/wiki/Artificial_intelligence

C Language http://en.wikipedia.org/wiki/C_(programming_language)

Objective-C http://en.wikipedia.org/wiki/Objective-C

Lisp Language http://en.wikipedia.org/wiki/Lisp_(programming_language)

Idea http://en.wikipedia.org/wiki/Idea

Concept http://en.wikipedia.org/wiki/Concept

Memory http://en.wikipedia.org/wiki/Memory

Mind http://en.wikipedia.org/wiki/Mind

Meme http://en.wikipedia.org/wiki/Meme

Cell http://en.wikipedia.org/wiki/Cell

Soul http://en.wikipedia.org/wiki/Soul

Understanding http://en.wikipedia.org/wiki/Understanding

Consciousness http://en.wikipedia.org/wiki/Consciousness

Basic English http://en.wikipedia.org/wiki/Basic_english

Word List http://en.wiktionary.org/wiki/Appendix:Basic_English_word_list

Chinese Room http://en.wikipedia.org/wiki/Chinese_room

Mind Uploading http://en.wikipedia.org/wiki/Mind_uploading

Internet Bots http://en.wikipedia.org/wiki/Internet_bot

AIML http://en.wikipedia.org/wiki/AIML

Natural Language

Conceptive C uses concepts to program natural language and Artifical Intelligence based computer language based on Objective C.

One of the first computer programs that I saw was in an Advert for the Apple II. Someone was typing in questions and the computer was answering them. Having a conversation with the computer seemed like an easy thing to do, only it's not.

Computers still have problems understanding English or Natural language. It didn't matter. I got hooked on programming computers. First in Basic, then 6502 Assembler, then Forth and C language.

I have always wanted to write a computer program that I could have a conversation with, I have thought about it over the years and I think I am a lot nearer to having a computer program that can understand English.

In a sort of you can't get there from here, I figured out that I needed to make a language that would allow me to program AI ideas and concepts. I have done a bit of object programming using Objective C. I liked the way that Objective C added just enough to C to allow for programming of Objects.

I wanted to do the same thing using Objective C to program idea's and concepts. Primarily I was thinking of using Conceptive C to program AI or Natural Language problems. The first program would be able to understand Basic English.

Basic English

Basic English (British American Scientific International Commercial) is a constructed (made-up) language to explain complex thoughts with 850 basic English words chosen by Charles Kay Ogden.

So I am looking to have a program that understands 850 Basic English words.

What is an idea? What s a concept? How do we understand something? What is meaning? How do people think about things?

If you take English you can break it down into words and sentences.

Sentences are built using words. Words have meanings. Words can be nouns, pronouns, verbs, adjectives or adverbs.

Can we represent knowledge using words in a way that a computer can use to understand the meaning of words being expressed.

A baby starts with very few words that have limited meaning. Most parents are happy enough once the baby can say one word "Mommy" or "Daddy" and it attach to mean that person.

If that is all the language that we learned it would be al that useful. Small children learn new words all the time and by the time they are say five years old they may already know 1,000 to 1,500 words and speak in sentences.

A definition of a word will use other words, which will each have definitions.

If we lookup word Idea, we get:

Definition of Idea: noun, a concept or mental impression.

So if we now lookup Concept, we get:

Definition of Concept: noun, an abstract idea.

Definition of Abstract: adjective, existing in thought or as an idea but not having a physical or concrete existence.

Definition of Noun: a word (other than a pronoun) used to identify any of a class of people, places, or things (common noun), or to name a particular one of these (proper noun).

Definition of Adjective: a word or phrase naming an attribute, added to or grammatically related to a noun to modify or describe it.

The point I am making is the Idea definition uses concept to describe itself and the definition of Concept uses idea to describe itself.

I'm using the Apple dictionary for the definitions. Idea actually had 3 definitions:

1. a thought or suggestion as to a possible course of action.

(the idea) the aim or purpose.

Philosophy (in Platonic thought) an eternally existing pattern of which individual things in any class are imperfect copies.

Word Definition

How do we get a definition that a computer can use to understand what a word means?

It seems like a problem if all words are defined with other words, that may not get us any meaning but lead us into circles of frustration.

It's more akin to how do we think or how do we understand the mean of a word.

Let's take a simpler word like Cat.

Definition of Cat:

noun

1. a small domesticated carnivorous mammal with soft fur, a short snout, and retractile claws. It is widely kept as a pet or for catching mice, and many breeds have been developed.

2 informal (particularly among jazz enthusiasts) a person, esp. a man.

Here are two different meanings a small furry anima, most people would mean this.

If you said “I have a cat” I would know what you mean and maybe even picture a cat in my minds eye.

Let’s look at Mind:

Definition of Mind:

noun

1 the element of a person that enables them to be aware of the world and their experiences, to think, and to feel; the faculty of consciousness and thought: as the thoughts ran through his mind, he came to a conclusion | people have the price they are prepared to pay settled in their minds.

• a person's mental processes contrasted with physical action: I wrote a letter in my mind.

2 a person's intellect: his keen mind.

• a person's memory: the company's name slips my mind .

• a person identified with their intellectual faculties: he was one of the greatest minds of his time.

3 a person's attention: I expect my employees to keep their minds on the job.

the will or determination to achieve something: anyone can lose weight if they set their mind to it.

verb [with obj.]

1 [often with negative] be distressed, annoyed, or worried by: I don't mind the rain.

• have an objection to: what does that mean, if you don't mind my asking? | [with clause] : do you mind if I have a cigarette?

• [with negative or in questions] (mind doing something) be reluctant to do something (often used in polite requests): I don't mind admitting I was worried.

• (would not mind something) informal used to express one's strong enthusiasm for something: I wouldn't mind some coaching from him!

2 regard as important; feel concern about: never mind the opinion polls | [no obj.] : why should she mind about a few snubs from people she didn't care for?

• [with clause in imperative] dated used to urge someone to remember or take care to bring about something: mind you look after the children.

• [no obj.] (also mind you) used to introduce a qualification to a previous statement: we've got some decorations up—not a lot, mind you.

• [no obj.] informal used to make a command more insistent or to draw attention to a statement: be early to bed tonight, mind.

• be obedient to: you think about how much Cal does for you, and you mind her, you hear?

• Scottish: I mind the time when he lost his false teeth.

3 take care of temporarily: we left our husbands to mind the children while we went out.

• [in imperative] used to warn someone to avoid injury or damage from a hazard: mind your head on that cupboard!

• [in imperative] be careful about the quality or nature of: mind your manners!

4 [with infinitive] (be minded) chiefly formal be inclined or disposed to do a particular thing: he was minded to reject the application | the Board was given leave to object if it was so minded.

Mind can be a noun or a verb. That's quite a complex definition. If I read it as a person I have a context and understanding of many word already in place. So it might mean something to me.

A computer would need a file of the text, which it would scan into memory. It could quite easily parse the text into words. Even group the words into sentences. But know what a word means or understand what a sentence is about and we start getting into some complex programming.

A problem is we have a mind, we understand thing, know what words mean, but how do we describe these things in ways that they can be replicated by a computer.

What we do have is computers can store something in a variable or represent things symbolically.

But I can look at my pet cat and know it's a cat, a small black cat or little Lion.

Visual recognition for computers will come at some time, in fact I'm sure there are some very good programs that can do that now.

For the moment I am sticking to words, so what I want is something like in a question and answer situation, if I ask a computer:

Q: What is a Cat?

A: A small domesticated carnivorous mammal with soft fur, a short snout, and retractile claws.

If I get that, that's fine. To do that the computer does not need to be conscious or have a real understanding of what a cat is or even have seen a real cat. If I can get it to do those things too, that would be a bonus.

So why am I calling this language Conceptive C?

Well I am trying to program using concepts to allow a computer to get some understanding or meaning about what it is talking about and be able to create new sentences that make sense. So the computer can have a conversation.

The main building block of Conceptive C, is a concept or an abstract idea.

Going back to the Apple Dictionary:

concept |□ kän□ sept|

noun

an abstract idea; a general notion: structuralism is a difficult concept | the concept of justice.

• a plan or intention; a conception: the center has kept firmly to its original concept.

• an idea or invention to help sell or publicize a commodity: a new concept in corporate hospitality.

Philosophy an idea or mental picture of a group or class of objects formed by combining all their aspects.

How do we implement that in code? Is it a new kind of object?

In “Object Oriented Programming” by Brad J. Cox, one of the inventors of Objective-C, he describes the counterparts of objective programming in conventional operator programming as:

object	a block of data
object id	pointer to block of data
method	apply function to data

What I need from Conceptual programming is the ability to learn or re-define itself. A concept is not fixed it can change and be updated, by new data and facts that don’t fit the original model that we had when we started programming the problem.

Machine Learning

I am talking about machine learning. This is more akin to an interpreter, rather than a compiler. Let’s look at these definitions:

compile |kəm□ pīl|

verb [with obj.]

1. produce (something, esp. a list, report, or book) by assembling information collected from other sources: the local authority must compile a list of taxpayers.

• collect (information) in order to produce something: the figures were compiled from a survey of 2,000 schoolchildren.

• accumulate (a specified score): the 49ers have compiled a league-leading 14–2 record.

2. Computing (of a computer) convert (a program) into a machine-code or lower-level form in which the program can be executed.

interpreter |in□tərpritər|

noun

a person who interprets, esp. one who translates speech orally.

Computing a program that can analyze and execute a program line by line.

Compiler Changes

In computing terms we have compilers and interpreters:

Compiler:

A computer program that can convert a program into a machine-code or lower-level form in which the program can be executed.

Interpreter:

A program that can analyze and execute a program line by line.

We have the concepts of run-time and compile-time. What happens when we run a program and what happens when we compile it. When we compile it , usually we get an executable file that we can run. When we run it we get something that we have programmed executing on the computer. An Interpreter takes our program and goes straight to the run stage, a line at a time.

How is this different with a compiled program that can do machine learning?

Well we compile and run our program as normal, but while the program is running if it decides that something meets certain parameters, the program will compile new code into its executable, to allow it to do a new function. This could result in the executable file or a new data file being added to the programs executable file.

Know Thyself

If we where really clever, we could write a program that knows nothing, but can learn everything. I don't think I'm that clever.

I do think one of the core concepts that we will need is the ability to know when we don't know something. We do need the ability to realize that

something is new and can be added to our program. Or for instance we currently have no apples. For instance:

Q: How many Apples do you have?

A: I have no Apples.

Q: I have given you an Apple how many Apples do you have now?

Q: I have one Apple.

This is like an empty object that would still have knowledge about what kind of object it might be, but a value of zero. Or maybe we owe someone 5 Apples, so we would know what an Apple is, know that we have no Apples and know that we need 5 Apples to repay a debt.

AI

Artificial intelligence (AI) is the intelligence of machines and the branch of computer science that aims to create it. AI textbooks define the field as "the study and design of intelligent agents" where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. John McCarthy, who coined the term in 1956, defines it as "the science and engineering of making intelligent machines."

The field was founded on the claim that a central property of humans, intelligence—the sapience of Homo sapiens—can be so precisely described that it can be simulated by a machine. This raises philosophical issues about the nature of the mind and the ethics of creating artificial beings, issues which have been addressed by myth, fiction and philosophy since antiquity. Artificial intelligence has been the subject of optimism, but has also suffered setbacks and, today, has become an essential part of the technology industry, providing the heavy lifting for many of the most difficult problems in computer science.

AI research is highly technical and specialized, deeply divided into subfields that often fail in the task of communicating with each other. Subfields have grown up around particular institutions, the work of individual researchers, the solution of specific problems, longstanding differences of opinion about how AI should be done and the application of widely differing tools. The central problems of AI include such traits as reasoning, knowledge, planning, learning, communication, perception and the ability to move and manipulate objects. General intelligence (or "strong AI") is still among the field's long term goals.

History

Thinking machines and artificial beings appear in Greek myths, such as Talos of Crete, the bronze robot of Hephaestus, and Pygmalion's Galatea. Human likenesses believed to have intelligence were built in every major civilization: animated cult images were worshipped in Egypt and Greece and humanoid automatons were built by Yan Shi, Hero of Alexandria and Al-Jazari. It was also widely believed that artificial beings had been created by Jābir ibn Hayyān, Judah Loew and Paracelsus. By the 19th and 20th centuries, artificial beings had become a common feature in fiction, as in Mary Shelley's Frankenstein or Karel Čapek's R.U.R. (Rossum's Universal Robots). Pamela McCorduck argues that all of these are examples of an ancient urge, as she describes it, "to forge the gods". Stories of these creatures and their fates discuss many of the same hopes, fears and ethical concerns that are presented by artificial intelligence.

Mechanical or "formal" reasoning has been developed by philosophers and mathematicians since antiquity. The study of logic led directly to the invention of the programmable digital electronic computer, based on the work of mathematician Alan Turing and others. Turing's theory of computation suggested that a machine, by shuffling symbols as simple as "0" and "1", could

simulate any conceivable act of mathematical deduction. This, along with concurrent discoveries in neurology, information theory and cybernetics, inspired a small group of researchers to begin to seriously consider the possibility of building an electronic brain.

The field of AI research was founded at a conference on the campus of Dartmouth College in the summer of 1956. The attendees, including John McCarthy, Marvin Minsky, Allen Newell and Herbert Simon, became the leaders of AI research for many decades. They and their students wrote programs that were, to most people, simply astonishing: Computers were solving word problems in algebra, proving logical theorems and speaking English. By the middle of the 1960s, research in the U.S. was heavily funded by the Department of Defense and laboratories had been established around the world. AI's founders were profoundly optimistic about the future of the new field: Herbert Simon predicted that "machines will be capable, within twenty years, of doing any work a man can do" and Marvin Minsky agreed, writing that "within a generation ... the problem of creating 'artificial intelligence' will substantially be solved".

They had failed to recognize the difficulty of some of the problems they faced. In 1974, in response to the criticism of Sir James Lighthill and ongoing pressure from the US Congress to fund more productive projects, both the U.S. and British governments cut off all undirected exploratory research in AI. The next few years, when funding for projects was hard to find, would later be called the "AI winter".

In the early 1980s, AI research was revived by the commercial success of expert systems, a form of AI program that simulated the knowledge and analytical skills of one or more human experts. By 1985 the market for AI had reached over a billion dollars. At the same time, Japan's fifth generation computer project inspired the U.S and British governments to restore funding for academic research in the field.

However, beginning with the collapse of the Lisp Machine market in 1987, AI once again fell into disrepute, and a second, longer lasting AI winter began.

In the 1990s and early 21st century, AI achieved its greatest successes, albeit somewhat behind the scenes. Artificial intelligence is used for logistics, data mining, medical diagnosis and many other areas throughout the technology industry. The success was due to several factors: the increasing computational power of computers (see Moore's law), a greater emphasis on solving specific subproblems, the creation of new ties between AI and other fields working on similar problems, and a new commitment by researchers to solid mathematical methods and rigorous scientific standards.

On 11 May 1997, Deep Blue became the first computer chess-playing system to beat a reigning world chess champion, Garry Kasparov. In 2005, a Stanford robot won the DARPA Grand Challenge by driving autonomously for 131 miles along an unrehearsed desert trail. Two years later, a team from CMU

won the DARPA Urban Challenge by autonomously navigating 55 miles in an Urban environment while adhering to traffic hazards and all traffic laws. In February 2011, in a Jeopardy! quiz show exhibition match, IBM's question answering system, Watson, defeated the two greatest Jeopardy! champions, Brad Rutter and Ken Jennings, by a significant margin.

The leading-edge definition of artificial intelligence research is changing over time. One pragmatic definition is: "AI research is that which computing scientists do not know how to do cost-effectively today." For example, in 1956 optical character recognition (OCR) was considered AI, but today, sophisticated OCR software with a context-sensitive spell checker and grammar checker software comes for free with most image scanners. No one would any longer consider already-solved computing science problems like OCR "artificial intelligence" today.

Low-cost entertaining chess-playing software is commonly available for tablet computers. DARPA no longer provides significant funding for chess-playing computing system development. The Kinect which provides a 3D body–motion interface for the Xbox 360 uses algorithms that emerged from lengthy AI research, but few consumers realize the technology source.

AI applications are no longer the exclusive domain of Department of defense R&D, but are now common place consumer items and inexpensive intelligent toys.

In common usage, the term "AI" no longer seems to apply to off-the-shelf solved computing-science problems, which may have originally emerged out of years of AI research.

Deduction, Reasoning, Problem Solving

"Can a machine act intelligently?" is still an open problem. Taking "A machine can act intelligently" as a working hypothesis, many researchers have attempted to build such a machine.

The general problem of simulating (or creating) intelligence has been broken down into a number of specific sub-problems. These consist of particular traits or capabilities that researchers would like an intelligent system to display. The traits described below have received the most attention.

Early AI researchers developed algorithms that imitated the step-by-step reasoning that humans use when they solve puzzles or make logical deductions. By the late 1980s and '90s, AI research had also developed highly successful methods for dealing with uncertain or incomplete information, employing concepts from probability and economics.

For difficult problems, most of these algorithms can require enormous computational resources — most experience a "combinatorial explosion": the amount of memory or computer time required becomes astronomical when the

problem goes beyond a certain size. The search for more efficient problem solving algorithms is a high priority for AI research.

Human beings solve most of their problems using fast, intuitive judgments rather than the conscious, step-by-step deduction that early AI research was able to model. AI has made some progress at imitating this kind of "sub-symbolic" problem solving: embodied agent approaches emphasize the importance of sensorimotor skills to higher reasoning; neural net research attempts to simulate the structures inside human and animal brains that give rise to this skill.

Knowledge Representation

An ontology represents knowledge as a set of concepts within a domain and the relationships between those concepts.

Main articles: Knowledge representation and Commonsense knowledge

Knowledge representation and knowledge engineering are central to AI research. Many of the problems machines are expected to solve will require extensive knowledge about the world. Among the things that AI needs to represent are: objects, properties, categories and relations between objects; situations, events, states and time; causes and effects; knowledge about knowledge (what we know about what other people know); and many other, less well researched domains. A representation of "what exists" is an ontology (borrowing a word from traditional philosophy), of which the most general are called upper ontologies.

Among the most difficult problems in knowledge representation are:

Commonsense Knowledge

Many of the things people know take the form of "working assumptions." For example, if a bird comes up in conversation, people typically picture an animal that is fist sized, sings, and flies. None of these things are true about all birds. John McCarthy identified this problem in 1969 as the qualification problem: for any commonsense rule that AI researchers care to represent, there tend to be a huge number of exceptions. Almost nothing is simply true or false in the way that abstract logic requires. AI research has explored a number of solutions to this problem.

The number of atomic facts that the average person knows is astronomical. Research projects that attempt to build a complete knowledge base of commonsense knowledge (e.g., Cyc) require enormous amounts of laborious ontological engineering — they must be built, by hand, one complicated concept at a time. A major goal is to have the computer understand enough concepts to be able to learn by reading from sources like the internet, and thus be able to add to its own ontology.

Much of what people know is not represented as "facts" or "statements" that they could express verbally. For example, a chess master will avoid a particular

chess position because it "feels too exposed" or an art critic can take one look at a statue and instantly realize that it is a fake. These are intuitions or tendencies that are represented in the brain non-consciously and sub-symbolically. Knowledge like this informs, supports and provides a context for symbolic, conscious knowledge. As with the related problem of sub-symbolic reasoning, it is hoped that situated AI or computational intelligence will provide ways to represent this kind of knowledge.

Learning

Machine learning has been central to AI research from the beginning. In 1956, at the original Dartmouth AI summer conference, Ray Solomonoff wrote a report on unsupervised probabilistic machine learning: "An Inductive Inference Machine".

Unsupervised learning is the ability to find patterns in a stream of input. Supervised learning includes both classification and numerical regression. Classification is used to determine what category something belongs in, after seeing a number of examples of things from several categories. Regression takes a set of numerical input/output examples and attempts to discover a continuous function that would generate the outputs from the inputs. In reinforcement learning the agent is rewarded for good responses and punished for bad ones. These can be analyzed in terms of decision theory, using concepts like utility. The mathematical analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory.

Natural Language Processing

A parse tree represents the syntactic structure of a sentence according to some formal grammar. Natural language processing gives machines the ability to read and understand the languages that humans speak. Many researchers hope that a sufficiently powerful natural language processing system would be able to acquire knowledge on its own, by reading the existing text available over the internet. Some straightforward applications of natural language processing include information retrieval (or text mining) and machine translation.

Creativity

A sub-field of AI addresses creativity both theoretically (from a philosophical and psychological perspective) and practically (via specific implementations of systems that generate outputs that can be considered creative, or systems that identify and assess creativity). A related area of computational research is Artificial intuition and Artificial imagination.

General Intelligence

Most researchers hope that their work will eventually be incorporated into a machine with general intelligence (known as strong AI), combining all the skills above and exceeding human abilities at most or all of them. A few

believe that anthropomorphic features like artificial consciousness or an artificial brain may be required for such a project.

Many of the problems above are considered AI-complete: to solve one problem, you must solve them all. For example, even a straightforward, specific task like machine translation requires that the machine follow the author's argument (reason), know what is being talked about (knowledge), and faithfully reproduce the author's intention (social intelligence). Machine translation, therefore, is believed to be AI-complete: it may require strong AI to be done as well as humans can do it.

Evaluating progress

In 1950, Alan Turing proposed a general procedure to test the intelligence of an agent now known as the Turing test. This procedure allows almost all the major problems of artificial intelligence to be tested. However, it is a very difficult challenge and at present all agents fail.

Artificial intelligence can also be evaluated on specific problems such as small problems in chemistry, hand-writing recognition and game-playing. Such tests have been termed subject matter expert Turing tests. Smaller problems provide more achievable goals and there are an ever-increasing number of positive results.

The broad classes of outcome for an AI test are: (1) Optimal: it is not possible to perform better. (2) Strong super-human: performs better than all humans. (3) Super-human: performs better than most humans. (4) Sub-human: performs worse than most humans. For example, performance at draughts is optimal, performance at chess is super-human and nearing strong super-human (see Computer chess#Computers versus humans) and performance at many everyday tasks (such as recognizing a face or crossing a room without bumping into something) is sub-human.

Philosophy

Artificial intelligence, by claiming to be able to recreate the capabilities of the human mind, is both a challenge and an inspiration for philosophy. Are there limits to how intelligent machines can be? Is there an essential difference between human intelligence and artificial intelligence? Can a machine have a mind and consciousness? The philosophy of artificial intelligence attempts to answer such questions as:

Can a machine act intelligently? Can it solve any problem that a person would solve by thinking?

Can a machine have a mind, mental states and consciousness in the same sense humans do? Can it feel?

Are human intelligence and machine intelligence the same? Is the human brain essentially a computer?

C Language

C (pronounced like the letter C) is a general-purpose computer programming language developed between 1969 and 1973 by Dennis Ritchie at the Bell Telephone Laboratories for use with the Unix operating system. Although C was designed for implementing system software, it is also widely used for developing portable application software.

C is one of the most widely used programming languages of all time and there are very few computer architectures for which a C compiler does not exist. C has greatly influenced many other popular programming languages, most notably C++, which began as an extension to C.

Design

C is an imperative (procedural) systems implementation language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. C was therefore useful for many applications that had formerly been coded in assembly language.

Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant and portably written C program can be compiled for a very wide variety of computer platforms and operating systems with few changes to its source code. The language has become available on a very wide range of platforms, from embedded microcontrollers to supercomputers.

Characteristics

Like most imperative languages in the ALGOL tradition, C has facilities for structured programming and allows lexical variable scope and recursion, while a static type system prevents many unintended operations. In C, all executable code is contained within subroutines, which are called "functions" (although not in the strict sense of functional programming). Function parameters are always passed by value. Pass-by-reference is simulated in C by explicitly passing pointer values. C program source text is free-format, using the semicolon as a statement terminator and curly braces for delimiting block.

C also exhibits the following more specific characteristics:

A small and fixed number of keywords, including a full set of flow of control primitives: for, if, while, switch and repeat loop (implemented as do..while). There is basically one namespace, and user-defined names are not distinguished from keywords by any kind of sigil.

A large number of compound arithmetical and logical operators, such as +=, -=, *=, ++, etc.

More than one assignment may be performed in a statement.

Function return values may be freely discarded.

Static, but weakly-enforced, typing; all data has a type, but implicit conversions can be performed, for instance, characters can be used as integers.

Definition follows use. C has no "define" keyword; instead, a statement beginning with the name of a type is taken as a definition. There is no "function" keyword; instead, the parentheses of an argument list indicate that a function is being defined.

User-defined (typedef) and compound types are possible.

Heterogeneous aggregate data types (struct) allow related data elements to be combined and manipulated as a unit.

Array indexing as a secondary notion, defined in terms of pointer arithmetic. Unlike structs, arrays are not first-class objects, and must be assigned and compared with library functions. There is no "array" keyword, in use or definition; instead, square brackets indicate arrays syntactically, e.g. myarr[].

Enumerated types are possible with the enum keyword. They are not tagged, and are freely interconvertible with integers.

Strings are not a predefined data type, but usually implemented as arrays of characters, with the same need to use library functions. The length of a C string is not fixed, nor part of its type. Variable-length C-style strings are null-terminated.

Low-level access to computer memory by converting machine addresses to typed pointers, on which arithmetic can be performed.

Procedures (subroutines not returning values) are a special case of function, returning the dummy type void.

Functions may not be defined within the lexical scope of other functions. (Although this may be provided as an language extension by some compilers.)

Function and data pointers supporting ad hoc run-time polymorphism

A preprocessor for macro definition, source code file inclusion, and conditional compilation.

C is modular: files containing functions can be compiled and linked separately.

Functions are exported by default from the files containing them, whereas variables are local to the file containing them unless explicitly imported.

Complex functionality such as I/O, string manipulation, and mathematical functions consistently delegated to library routines.

The traditional C toolset consists of compilers, linkers, debuggers, etc invoked from the command line. Integrated development environments have also been introduced.

C does not include some features found in newer, more modern high-level languages, including:

Object orientation

Garbage collection

Type reflection

Early Developments

The initial development of C occurred at AT&T Bell Labs between 1969 and 1973; according to Ritchie, the most creative period occurred in 1972. It was named "C" because its features were derived from an earlier language called "B", which according to Ken Thompson was a stripped-down version of the BCPL programming language.

The origin of C is closely tied to the development of the Unix operating system, originally implemented in assembly language on a PDP-7 by Ritchie and Thompson, incorporating several ideas from colleagues. Eventually they decided to port the operating system to a PDP-11. B's inability to take advantage of some of the PDP-11's features, notably byte addressability, led to the development of an early version of C.

The original PDP-11 version of the Unix system was developed in assembly language. By 1973, with the addition of struct types, the C language had become powerful enough that most of the Unix kernel was rewritten in C. This was one of the first operating system kernels implemented in a language other than assembly. (Earlier instances include the Multics system (written in PL/I), and MCP (Master Control Program) for the Burroughs B5000 written in ALGOL in 1961.)

K&R C

In 1978, Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language. This book, known to C programmers as "K&R", served for many years as an informal specification of the language. The version of C that it describes is commonly referred to as K&R C. The second edition of the book covers the later ANSI C standard.

K&R introduced several language features:

standard I/O library

long int data type

unsigned int data type

compound assignment operators of the form =op (such as =-) were changed to the form op= to remove the semantic ambiguity created by such constructs as i=-10, which had been interpreted as i =- 10 instead of the possibly intended i = -10

Even after the publication of the 1989 C standard, for many years K&R C was still considered the "lowest common denominator" to which C programmers restricted themselves when maximum portability was desired, since many older compilers were still in use, and because carefully written K&R C code can be legal Standard C as well.

In early versions of C, only functions that returned a non-int value needed to be declared if used before the function definition; a function used without any previous declaration was assumed to return type int, if its value was used.

For example:

```
long some_function();
/* int */ other_function();
/* int */ calling_function()
{
long test1;
register /* int */ test2;
 test1 = some_function();
if (test1 > 0)
  test2 = 0;
else
  test2 = other_function();
return test2;
}
```

All the above commented-out int declarations could be omitted in K&R C.

Since K&R function declarations did not include any information about function arguments, function parameter type checks were not performed, although some compilers would issue a warning message if a local function was called with the wrong number of arguments, or if multiple calls to an external function used different numbers or types of arguments. Separate tools such as Unix's lint utility were developed that (among other things) could check for consistency of function use across multiple source files.

In the years following the publication of K&R C, several unofficial features were added to the language, supported by compilers from AT&T and some other vendors. These included:

void functions (i.e. functions with no return value)

functions returning struct or union types (rather than pointers)

assignment for struct data types

enumerated types

The large number of extensions and lack of agreement on a standard library, together with the language popularity and the fact that not even the Unix compilers precisely implemented the K&R specification, led to the necessity of standardization.

ANSI C and ISO C

During the late 1970s and 1980s, versions of C were implemented for a wide variety of mainframe computers, minicomputers, and microcomputers, including the IBM PC, as its popularity began to increase significantly.

In 1983, the American National Standards Institute (ANSI) formed a committee, X3J11, to establish a standard specification of C. In 1989, the standard was ratified as ANSI X3.159-1989 "Programming Language C". This version of the language is often referred to as ANSI C, Standard C, or sometimes C89.

In 1990, the ANSI C standard (with formatting changes) was adopted by the International Organization for Standardization (ISO) as ISO/IEC 9899:1990, which is sometimes called C90. Therefore, the terms "C89" and "C90" refer to the same programming language.

ANSI, like other national standards bodies, no longer develops the C standard independently, but defers to the ISO C standard. National adoption of updates to the international standard typically occurs within a year of ISO publication.

One of the aims of the C standardization process was to produce a superset of K&R C, incorporating many of the unofficial features subsequently introduced. The standards committee also included several additional features such as function prototypes (borrowed from C++), void pointers, support for international character sets and locales, and preprocessor enhancements. Although the syntax for parameter declarations was augmented to include the style used in C++, the K&R interface continued to be permitted, for compatibility with existing source code.

C89 is supported by current C compilers, and most C code being written today is based on it. Any program written only in Standard C and without any hardware-dependent assumptions will run correctly on any platform with a conforming C implementation, within its resource limits. Without such precautions, programs may compile only on a certain platform or with a particular compiler, due, for example, to the use of non-standard libraries, such as GUI libraries, or to a reliance on compiler- or platform-specific attributes such as the exact size of data types and byte endianness.

In cases where code must be compilable by either standard-conforming or K&R C-based compilers, the __STDC__ macro can be used to split the code into Standard and K&R sections to prevent using on a K&R C-based compiler features available only in Standard C.

C99

After the ANSI/ISO standardization process, the C language specification remained relatively static for some time. In 1995 Normative Amendment 1 to the 1990 C standard was published, to correct some details and to add more extensive support for international character sets. The C standard was further revised in the late 1990s, leading to the publication of ISO/IEC 9899:1999 in 1999, which is commonly referred to as "C99". It has since been amended three times by Technical Corrigenda. The international C standard is maintained by the working group ISO/IEC JTC1/SC22/WG14.

C99 introduced several new features, including inline functions, several new data types (including long long int and a complex type to represent complex numbers), variable-length arrays, support for variadic macros (macros of variable arity) and support for one-line comments beginning with //, as in BCPL or C++. Many of these had already been implemented as extensions in several C compilers.

C99 is for the most part backward compatible with C90, but is stricter in some ways; in particular, a declaration that lacks a type specifier no longer has int implicitly assumed. A standard macro __STDC_VERSION__ is defined with value 199901L to indicate that C99 support is available. GCC, Sun Studio and other C compilers now support many or all of the new features of C99.

Embedded C

Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations.

In 2008, the C Standards Committee published a technical report extending the C language to address these issues by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as, fixed-point arithmetic, named address spaces, and basic I/O hardware addressing.

Uses

C is often used for "system programming", including implementing operating systems and embedded system applications, due to a combination of desirable characteristics such as code portability and efficiency, ability to access specific hardware addresses, ability to pun types to match externally imposed data access requirements, and low run-time demand on system resources. C can also be used for website programming using CGI as a "gateway" for information between the Web application, the server, and the browser. Some reasons for choosing C over interpreted languages are its speed, stability, and near-universal availability.

One consequence of C's wide acceptance and efficiency is that compilers, libraries, and interpreters of other programming languages are often implemented in C. The primary implementations of Python (CPython), Perl 5, and PHP are all written in C.

Due to its thin layer of abstraction and low overhead, C allows efficient implementations of algorithms and data structures, which is useful for programs that perform a lot of computations. For example, the GNU Multi-Precision Library, the GNU Scientific Library, Mathematica and MATLAB are completely or partially written in C.

C is sometimes used as an intermediate language by implementations of other languages. This approach may be used for portability or convenience; by using C as an intermediate language, it is not necessary to develop machine-specific code generators. Some languages and compilers which have used C this way are BitC, C++, COBOL, Eiffel, Gambit, GHC, Squeak, and Vala. However, C was designed as a programming language, not as a compiler target language, and is thus less than ideal for use as an intermediate language. This has led to development of C-based intermediate languages such as C--.

C has also been widely used to implement end-user applications, but much of that development has shifted to newer languages.

Syntax

C has a formal grammar specified by the C standard. Unlike languages such as FORTRAN 77, C source code is free-form which allows arbitrary use of whitespace to format code, rather than column-based or text-line-based restrictions. Comments may appear either between the delimiters /* and */, or (in C99) following // until the end of the line.

C source files contain declarations and function definitions. Function definitions, in turn, contain declarations and statements. Declarations either define new types using keywords such as struct, union, and enum, or assign types to and perhaps reserve storage for new variables, usually by writing the type followed by the variable name. Keywords such as char and int specify built-in types. Sections of code are enclosed in braces ({ and }, sometimes called "curly brackets") to limit the scope of declarations and to act as a single statement for control structures.

As an imperative language, C uses statements to specify actions. The most common statement is an expression statement, consisting of an expression to be evaluated, followed by a semicolon; as a side effect of the evaluation, functions may be called and variables may be assigned new values. To modify the normal sequential execution of statements, C provides several control-flow statements identified by reserved keywords. Structured programming is supported by if(-else) conditional execution and by do-while, while, and for iterative execution (looping). The for statement has separate initialization, testing, and reinitialization expressions, any or all of which can be omitted. break and continue can be used to leave the innermost enclosing loop

statement or skip to its reinitialization. There is also a non-structured goto statement which branches directly to the designated label within the function. switch selects a case to be executed based on the value of an integer expression.

Expressions can use a variety of built-in operators (see below) and may contain function calls. The order in which arguments to functions and operands to most operators are evaluated is unspecified. The evaluations may even be interleaved. However, all side effects (including storage to variables) will occur before the next "sequence point"; sequence points include the end of each expression statement, and the entry to and return from each function call. Sequence points also occur during evaluation of expressions containing certain operators (&&, ||, ?: and the comma operator). This permits a high degree of object code optimization by the compiler, but requires C programmers to take more care to obtain reliable results than is needed for other programming languages.

Keywords

C89 has 32 keywords (reserved words with special meaning):

auto double int struct

break else long switch

case enum register typedef

char extern return union

const float short unsigned

continue for signed void

default goto sizeof volatile

do if static while

C99 adds five more keywords:

_Bool inline

_Complex restrict

_Imaginary

C1X adds seven more keywords:

_Alignas _Generic _Static_assert

_Alignof _Noreturn _Thread_local

_Atomic

Operators

C supports a rich set of operators, which are symbols used within an expression to specify the manipulations to be performed while evaluating that expression. C has operators for:

arithmetic: +, -, *, /, %

assignment: =

augmented assignment: +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=

bitwise logic: ~, &, |, ^

bitwise shifts: <<, >>

boolean logic: !, &&, ||

conditional evaluation: ? :

equality testing: ==, !=

calling functions: ( )

increment and decrement: ++ and --

member selection: ., ->

object size: sizeof

order relations: <, <=, >, >=

reference and dereference: &, *, [ ]

sequencing: ,

subexpression grouping: ( )

type conversion: (typename)

"Hello, world" Example

The "hello, world" example, which appeared in the first edition of K&R, has become the model for an introductory program in most programming textbooks, regardless of programming language. The program prints "hello, world" to the standard output, which is usually a terminal or screen display.

The original version was:

```
main()
{
printf("hello, world\n");
}
```

A standard-conforming "hello, world" program is:

```
#include <stdio.h>
int main(void)
```

```
{
printf("hello, world\n");
return 0;
}
```

The first line of the program contains a preprocessing directive, indicated by #include. This causes the compiler to replace that line with the entire text of the stdio.h standard header, which contains declarations for standard input and output functions such as printf. The angle brackets surrounding stdio.h indicate that stdio.h is located using a search strategy that prefers standard headers to other headers having the same name. (Double quotes are used to include local or project-specific header files.)

The next line indicates that a function named main is being defined. The main function serves a special purpose in C programs; the run-time environment calls the main function to begin program execution. The type specifier int indicates that the value that is returned to the invoker (in this case the run-time environment) as a result of evaluating the main function, is an integer. The keyword void as a parameter list indicates that this function takes no arguments.

The opening curly brace indicates the beginning of the definition of the main function.

The next line calls (diverts execution to) a function named printf, which is supplied from a system library. In this call, the printf function is passed (provided with) a single argument, the address of the first character in the string literal "hello, world\n". The string literal is an unnamed array with elements of type char, set up automatically by the compiler with a final 0-valued character to mark the end of the array (printf needs to know this). The \ n is an escape sequence that C translates to a newline character, which on output signifies the end of the current line. The return value of the printf function is of type int, but it is silently discarded since it is not used. (A more careful program might test the return value to determine whether or not the printf function succeeded.) The semicolon ; terminates the statement.

The return statement terminates the execution of the main function and causes it to return the integer value 0, which is interpreted by the run-time system as an exit code indicating successful execution.

The closing curly brace indicates the end of the code for the main function.

Data Types

C has a static weak typing type system that shares some similarities with that of other ALGOL descendants such as Pascal. There are built-in types for integers of various sizes, both signed and unsigned, floating-point numbers, characters, and enumerated types (enum). C99 added a boolean datatype. There

are also derived types including arrays, pointers, records (struct), and untagged unions (union).

C is often used in low-level systems programming where escapes from the type system may be necessary. The compiler attempts to ensure type correctness of most expressions, but the programmer can override the checks in various ways, either by using a type cast to explicitly convert a value from one type to another, or by using pointers or unions to reinterpret the underlying bits of a value in some other way.

Pointers

C supports the use of pointers, a type of reference that records the address or location of an object or function in memory. Pointers can be dereferenced to access data stored at the address pointed to, or to invoke a pointed-to function. Pointers can be manipulated using assignment or pointer arithmetic. The run-time representation of a pointer value is typically a raw memory address (perhaps augmented by an offset-within-word field), but since a pointer's type includes the type of the thing pointed to, expressions including pointers can be type-checked at compile time. Pointer arithmetic is automatically scaled by the size of the pointed-to data type. (See Array-pointer interchangeability below.) Pointers are used for many different purposes in C. Text strings are commonly manipulated using pointers into arrays of characters. Dynamic memory allocation is performed using pointers. Many data types, such as trees, are commonly implemented as dynamically allocated struct objects linked together using pointers. Pointers to functions are useful for passing functions as arguments to higher-order functions (such as qsort or bsearch) or as callbacks to be invoked by event handlers.

A null pointer is a pointer that explicitly points to no valid location. It is created by setting a pointer to the integer value zero. The internal representation of pointers is not specified, and null pointers might not be represented in the same way as a zero integer. Dereferencing a null pointer is undefined, often resulting in a segmentation fault. Null pointers are useful for indicating special cases such as no next pointer in the final node of a linked list, or as an error indication from functions returning pointers. Null pointers are sometimes represented by a standard macro named NULL. Null pointers logically evaluate to false, while all other pointers which evaluate to true.

Void pointers (void *) point to objects of unknown type, and can therefore be used as "generic" data pointers. Since the size and type of the pointed-to object is not known, void pointers cannot be dereferenced, nor is pointer arithmetic on them allowed, although they can easily be (and in many contexts implicitly are) converted to and from any other object pointer type.

Careless use of pointers is potentially dangerous. Because they are typically unchecked, a pointer variable can be made to point to any arbitrary location, which can cause undesirable effects. Although properly-used pointers point to safe places, they can be made to point to unsafe places by using invalid pointer

arithmetic; the objects they point to may be deallocated and reused (dangling pointers); they may be used without having been initialized (wild pointers); or they may be directly assigned an unsafe value using a cast, union, or through another corrupt pointer. In general, C is permissive in allowing manipulation of and conversion between pointer types, although compilers typically provide options for various levels of checking. Some other programming languages address these problems by using more restrictive reference types.

Arrays

Array types in C are traditionally of a fixed, static size specified at compile time. (The more recent C99 standard also allows a form of variable-length arrays.) However, it is also possible to allocate a block of memory (of arbitrary size) at run-time, using the standard library's malloc function, and treat it as an array. C's unification of arrays and pointers (see below) means that true arrays and these dynamically-allocated, simulated arrays are virtually interchangeable. Since arrays are always accessed (in effect) via pointers, array accesses are typically not checked against the underlying array size, although the compiler may provide bounds checking as an option. Array bounds violations are therefore possible and rather common in carelessly written code, and can lead to various repercussions, including illegal memory accesses, corruption of data, buffer overruns, and run-time exceptions.

C does not have a special provision for declaring multidimensional arrays, but rather relies on recursion within the type system to declare arrays of arrays, which effectively accomplishes the same thing. The index values of the resulting "multidimensional array" can be thought of as increasing in row-major order.

Multidimensional arrays are commonly used in numerical algorithms (mainly from applied linear algebra) to store matrices. The structure of the C array is well suited to this particular task. However, since arrays are passed merely as pointers, the bounds of the array must be known fixed values or else explicitly passed to any subroutine that requires them, and dynamically sized arrays of arrays cannot be accessed using double indexing. (A workaround for this is to allocate the array with an additional "row vector" of pointers to the columns.)

C99 introduced "variable-length arrays" which address some, but not all, of the issues with ordinary C arrays.

Array-pointer Interchangeability

A distinctive (but potentially confusing) feature of C is its treatment of arrays and pointers. The array-subscript notation x[i] can also be used when x is a pointer; the interpretation (using pointer arithmetic) is to access the (i + 1)th object of several adjacent data objects pointed to by x, counting the object that x points to (which is x[0]) as the first element of the array.

Formally, x[i] is equivalent to *(x + i). Since the type of the pointer involved is known to the compiler at compile time, the address that x + i points to is not

the address pointed to by x incremented by i bytes, but rather incremented by i multiplied by the size of an element that x points to. The size of these elements can be determined with the operator sizeof by applying it to any dereferenced element of x, as in n = sizeof *x or n = sizeof x[0].

Furthermore, in most expression contexts (a notable exception is as operand of sizeof), the name of an array is automatically converted to a pointer to the array's first element; this implies that an array is never copied as a whole when named as an argument to a function, but rather only the address of its first element is passed. Therefore, although function calls in C use pass-by-value semantics, arrays are in effect passed by reference.

The number of elements in a declared array x can be determined as sizeof x / sizeof x[0].

An interesting demonstration of the interchangeability of pointers and arrays is shown below. The four assignments are equivalent and each is valid C code.

```
/* x is an array OR a pointer. i is an integer. */
x[i] = 1;          /* equivalent to *(x + i) */
*(x + i) = 1;
*(i + x) = 1;
i[x] = 1;          /* equivalent to *(i + x) */
```

Note that although all four assignments are equivalent, only the first represents good coding style. The last line might be found in obfuscated C code.

Despite this apparent equivalence between array and pointer variables, there is still a distinction to be made between them. Even though the name of an array is, in most expression contexts, converted into a pointer (to its first element), this pointer does not itself occupy any storage, unlike a pointer variable. Consequently, what an array "points to" cannot be changed, and it is impossible to assign a value to an array variable. (Array contents may be copied, however, e.g., by using the memcpy function, or by accessing the individual elements.)

Memory Management

One of the most important functions of a programming language is to provide facilities for managing memory and the objects that are stored in memory. C provides three distinct ways to allocate memory for objects:

Static memory allocation: space for the object is provided in the binary at compile-time; these objects have an extent (or lifetime) as long as the binary which contains them is loaded into memory.

Automatic memory allocation: temporary objects can be stored on the stack, and this space is automatically freed and reusable after the block in which they are declared is exited.

Dynamic memory allocation: blocks of memory of arbitrary size can be requested at run-time using library functions such as malloc from a region of memory called the heap; these blocks persist until subsequently freed for reuse by calling the library function realloc or free

These three approaches are appropriate in different situations and have various tradeoffs. For example, static memory allocation has little allocation overhead, automatic allocation may involve slightly more overhead, and dynamic memory allocation can potentially have a great deal of overhead for both allocation and deallocation. The persistent nature of static objects is useful for maintaining state information across function calls, automatic allocation is easy to use but stack space is typically much more limited and transient than either static memory or heap space, and dynamic memory allocation allows convenient allocation of objects whose size is known only at run-time. Most C programs make extensive use of all three.

Where possible, automatic or static allocation is usually simplest because the storage is managed by the compiler, freeing the programmer of the potentially error-prone chore of manually allocating and releasing storage. However, many data structures can change in size at runtime, and since static allocations (and automatic allocations before C99) must have a fixed size at compile-time, there are many situations in which dynamic allocation is necessary. Prior to the C99 standard, variable-sized arrays were a common example of this. (See the article on malloc for an example of dynamically allocated arrays.)

Unless otherwise specified, static objects contain zero or null pointer values upon program startup. Automatically and dynamically allocated objects are initialized only if an initial value is explicitly specified; otherwise they initially have indeterminate values (typically, whatever bit pattern happens to be present in the storage, which might not even represent a valid value for that type). If the program attempts to access an uninitialized value, the results are undefined. Many modern compilers try to detect and warn about this problem, but both false positives and false negatives can occur.

Another issue is that heap memory allocation has to be synchronized with its actual usage in any program in order for it to be reused as much as possible. For example, if the only pointer to a heap memory allocation goes out of scope or has its value overwritten before free() is called, then that memory cannot be recovered for later reuse and is essentially lost to the program, a phenomenon known as a memory leak. Conversely, it is possible to release memory too soon and continue to access it; however, since the allocation system can re-allocate or itself use the freed memory, unpredictable behavior is likely to occur. Typically, the symptoms will appear in a portion of the program far removed from the actual error, making it difficult to track down the problem. (Such issues are ameliorated in languages with automatic garbage collection.)

Libraries

The C programming language uses libraries as its primary method of extension. In C, a library is a set of functions contained within a single "archive" file. Each library typically has a header file, which contains the prototypes of the functions contained within the library that may be used by a program, and declarations of special data types and macro symbols used with these functions. In order for a program to use a library, it must include the library's header file, and the library must be linked with the program, which in many cases requires compiler flags (e.g., -lm, shorthand for "math library").

The most common C library is the C standard library, which is specified by the ISO and ANSI C standards and comes with every C implementation (“freestanding” [embedded] C implementations may provide only a subset of the standard library). This library supports stream input and output, memory allocation, mathematics, character strings, and time values.

Another common set of C library functions are those used by applications specifically targeted for Unix and Unix-like systems, especially functions which provide an interface to the kernel. These functions are detailed in various standards such as POSIX and the Single UNIX Specification.

Since many programs have been written in C, there are a wide variety of other libraries available. Libraries are often written in C because C compilers generate efficient object code; programmers then create interfaces to the library so that the routines can be used from higher-level languages like Java, Perl, and Python.

Objective C

Objective-C is a reflective, object-oriented programming language that adds Smalltalk-style messaging to the C programming language.

Today, it is used primarily on Apple's Mac OS X and iOS: two environments derived from the OpenStep standard, though not compliant with it. Objective-C is the primary language used for Apple's Cocoa API, and it was originally the main language on NeXT's NeXTSTEP OS. Generic Objective-C programs that do not use these libraries can also be compiled for any system supported by gcc or Clang.

History

Objective-C was created primarily by Brad Cox and Tom Love in the early 1980s at their company Stepstone. Both had been introduced to Smalltalk while at ITT Corporation's Programming Technology Center in 1981. The earliest work on Objective C traces back to around that time. Cox was intrigued by problems of true reusability in software design and programming. He realized that a language like Smalltalk would be invaluable in building development environments for system developers at ITT. However, he and Tom Love also recognized that backward compatibility with C was critically important in ITT's telecom engineering milieu. Cox began writing a pre-processor for C to add some of the capabilities of Smalltalk. He soon had a working implementation of an object-oriented extension to the C language, which he called "OOPC" for Object-Oriented Pre-Compiler. Love was hired by Schlumberger Research in 1982 and had the opportunity to acquire the first commercial copy of Smalltalk-80, which further influenced the development of their brainchild.

In order to demonstrate that real progress could be made, Cox showed that making interchangeable software components really needed only a few practical changes to existing tools. Specifically, they needed to support objects in a flexible manner, come supplied with a usable set of libraries, and allow for the code (and any resources needed by the code) to be bundled into a single cross-platform format.

Love and Cox eventually formed a new venture, Productivity Products International (PPI), to commercialize their product, which coupled an Objective-C compiler with class libraries. In 1986, Cox published the main description of Objective-C in its original form in the book Object-Oriented Programming, An Evolutionary Approach. Although he was careful to point out that there is more to the problem of reusability than just the language, Objective-C often found itself compared feature for feature with other languages.

Popularization through NeXT

After Steve Jobs left Apple Inc., he started the company NeXT. In 1988, NeXT licensed Objective-C from StepStone (the new name of PPI, the owner of the Objective-C trademark) and extended the GCC compiler to support Objective-C, and developed the AppKit and Foundation Kit libraries on which the NeXTstep user interface and interface builder were based. While the NeXT workstations failed to make a great impact in the marketplace, the tools were widely lauded in the industry. This led NeXT to drop hardware production and focus on software tools, selling NeXTstep (and OpenStep) as a platform for custom programming.

The GNU project started work on its free clone of NeXTStep, named GNUstep, based on the OpenStep standard. Dennis Glatting wrote the first GNU Objective-C runtime in 1992. The GNU Objective-C runtime, which has been in use since 1993, is the one developed by Kresten Krab Thorup when he was a university student in Denmark. Thorup also worked at NeXT from 1993 to 1996.

After acquiring NeXT in 1996, Apple Computer used OpenStep in its new operating system, Mac OS X. This included Objective-C and NeXT's Objective-C based developer tool, Project Builder (which had been expanded and is now called Xcode), as well as its interface design tool, Interface Builder. Most of Apple's present-day Cocoa API is based on OpenStep interface objects, and is the most significant Objective-C environment being used for active development.

Syntax

Objective-C is a thin layer on top of C, and moreover is a strict superset of C; it is possible to compile any C program with an Objective-C compiler, and to freely include C code within an Objective-C class.

Objective-C derives its object syntax from Smalltalk. All of the syntax for non-object-oriented operations (including primitive variables, preprocessing, expressions, function declarations, and function calls) are identical to that of C, while the syntax for object-oriented features is an implementation of Smalltalk-style messaging.

Messages

The Objective-C model of object-oriented programming is based on message passing to object instances. In Objective-C one does not call a method; one sends a message. This is unlike the Simula-style programming model used by C++. The difference between these two concepts is in how the code referenced by the method or message name is executed. In a Simula-style language, the method name is in most cases bound to a section of code in the target class by the compiler. In Smalltalk and Objective-C, the target of a message is resolved at runtime, with the receiving object itself interpreting the message. A method is identified by a selector or SEL — a NUL-terminated string representing its name — and resolved to a C method pointer implementing it: an IMP. A consequence of this is that the message-passing system has no type checking.

The object to which the message is directed — the receiver — is not guaranteed to respond to a message, and if it does not, it simply raises an exception.

Sending the message method to the object pointed to by the pointer obj would require the following code in C++:

obj->method(argument);

In Objective-C, this is written as follows:

[obj method: argument];

Both styles of programming have their strengths and weaknesses. Object-oriented programming in the Simula style allows multiple inheritances and faster execution by using compile-time binding whenever possible, but it does not support dynamic binding by default. It also forces all methods to have a corresponding implementation unless they are virtual, meaning the method is a placeholder for methods with the same name to be defined in objects derived from the base object. An implementation is still required for the method to be called in the derived object. Smalltalk-style programming allows messages to go unimplemented, with the method resolved to its implementation at runtime. For example, a message may be sent to a collection of objects, to which only some will be expected to respond, without fear of producing runtime errors. Message passing also does not require that an object be defined at compile time. (See the dynamic typing section below for more advantages of dynamic (late) binding.)

Due to the overhead of interpreting the messages, an initial Objective-C message takes three times as long as a C++ virtual method call. Subsequent calls are IMP cached and are faster than the C++ virtual method call in some implementations.

Interfaces and Implementations

Objective-C requires that the interface and implementation of a class be in separately declared code blocks. By convention, developers place the interface in a header file and the implementation in a code file. The header files, normally suffixed .h, are similar to C header files while the implementation (method) files, normally suffixed .m, can be very similar to C code files.

Interface

The interface of a class is usually defined in a header file. A common convention is to name the header file after the name of the class, e.g. Ball.h would contain the interface for the class Ball.

An interface declaration takes the form:

@interface classname : superclassname {

// instance variables

}

```
+ classMethod1;

+ (return_type)classMethod2;

+ (return_type)classMethod3:(param1_type)param1_varName;

- (return_type)instanceMethod1:(param1_type)param1_varName :
(param2_type)param2_varName;

- (return_type)instanceMethod2WithParameter:
(param1_type)param1_varName andOtherParameter:
(param2_type)param2_varName;

@end
```

In the above, plus signs denote class methods, or methods that can be called on the class itself (not on an instance), and minus signs denote instance methods, which can only be called on a particular instance of the class. Class methods also have no access to instance variables.

Return types can be any standard C type, a pointer to a generic Objective-C object, or a pointer to a specific type of object such as NSArray *, NSImage *, or NSString *. The default return type is the generic Objective-C type id.

Method arguments begin with a colon followed by the expected argument type in parentheses and the argument name. In some cases (e.g. when writing system APIs) it is useful to add descriptive text before each parameter.

```
- (void)setRangeStart:(int)start end:(int)end;

(void)importDocumentWithName:(NSString *)name
withSpecifiedPreferences:(Preferences *)prefs beforePage:(int)insertPage;
```

Implementation

The interface only declares the class interface and not the methods themselves: the actual code is written in the implementation file. Implementation (method) files normally have the file extension .m, which originally signified "messages".

```
@implementation classname

+ (return_type)classMethod {

// implementation

}

- (return_type)instanceMethod {

// implementation

}

@end
```

Methods are written using their interface declarations. Comparing Objective-C and C:

```
- (int)method:(int)i {
return [self square_root:i];
}
int function (int i) {
return square_root(i);
}
```

The syntax allows pseudo-naming of arguments.

```
- (int)changeColorToRed:(float)red green:(float)green blue:(float)blue;
[myColor changeColorToRed:5.0 green:2.0 blue:6.0];
```

Internal representations of a method vary between different implementations of Objective-C. If myColor is of the class Color, instance method -changeColorToRed:green:blue: might be internally labeled _i_Color_changeColorToRed_green_blue. The i is to refer to an instance method, with the class and then method names appended and colons changed to underscores. As the order of parameters is part of the method name, it cannot be changed to suit coding style or expression as with true named parameters.

However, internal names of the function are rarely used directly. Generally, messages are converted to function calls defined in the Objective-C runtime library. It is not necessarily known at link time which method will be called because the class of the receiver (the object being sent the message) need not be known until runtime.

Instantiation

Once an Objective-C class is written, it can be instantiated. This is done by first allocating an uninitialized instance of the class (an object) and then by initializing it. An object is not fully functional until both steps have been completed. These steps should be accomplished with a single line of code so that there is never an allocated object that hasn't undergone initialization (and because it is not advisable to keep the intermediate result since -init can return a different object than that which it is called on).

Instantiation with the default, no-parameter initializer:

```
MyObject *o = [[MyObject alloc] init];
```

Instantiation with a custom initializer:

```
MyObject *o = [[MyObject alloc] initWithString:myString];
```

In the case where no custom initialization is being performed, the "new" method can often be used in place of the alloc-init messages:

```
MyObject *o = [MyObject new];
```

The alloc message allocates enough memory to hold all the instance variables for an object, sets all the instance variables to zero values, and turns the memory into an instance of the class; at no point during the initialization is the memory an instance of the superclass.

The init message performs the set-up of the instance upon creation. The init method is often written as follows:

```
- (id)init {
self = [super init];
if (self) {
// perform initialization of object here
        }
return self;
}
```

In the above example, notice the id return type. This type stands for "pointer to any object" in Objective-C (See the Dynamic typing section).

The initializer pattern is used in order to assure that the object is properly initialized by its superclass before the init method performs its initialization. It performs the following actions:

1. self = [super init]

Sends the superclass instance an init message and assigns the result to self (pointer to the current object).

2. if (self)

Checks if the returned object pointer is valid before performing any initialization.

3. return self

Returns the value of self to the caller.

A non-valid object pointer has the value nil; conditional statements like "if" treat nil like a null pointer, so the initialization code will not be executed if [super init] returned nil. If there is an error in initialization the init method should perform any necessary cleanup, including sending a "release" message to self, and return nil to indicate that initialization failed, any checking for such errors must only be performed after having called the superclass initialization to ensure that destroying the object will be done correctly.

If a class has more than one initialization method, only one of them (the "dedicated initializer") needs to follow this pattern; others can call the dedicated initializer instead of the superclass initializer.

Protocols

Objective-C was extended at NeXT to introduce the concept of multiple inheritance of specification, but not implementation, through the introduction of protocols. This is a pattern achievable either as an abstract multiply inherited base class in C++, or as an "interface" (as in Java and C#). Objective-C makes use of ad hoc protocols called informal protocols and compiler-enforced protocols called formal protocols.

An informal protocol is a list of methods that a class can opt to implement. It is specified in the documentation, since it has no presence in the language. Informal protocols often include optional methods, which, if implemented, can change the behavior of a class. For example, a text field class might have a delegate that implements an informal protocol with an optional method for performing auto-completion of user-typed text. The text field discovers whether the delegate implements that method (via reflection) and, if so, calls the delegate's method to support the auto-complete feature.

A formal protocol is similar to an interface in Java or C#. It is a list of methods that any class can declare itself to implement. Versions of Objective-C before 2.0 required that a class must implement all methods in a protocol it declares itself as adopting; the compiler will emit an error if the class does not implement every method from its declared protocols. Objective-C 2.0 added support for marking certain methods in a protocol optional, and the compiler will not enforce implementation of optional methods.

The Objective-C concept of protocols is different from the Java or C# concept of interfaces, in that a class may implement a protocol without being declared to implement that protocol. The difference is not detectable from outside code. [dubious – discuss] Formal protocols cannot provide any implementations, they simply assure callers that classes that conform to the protocol will provide implementations. In the NeXT/Apple library, protocols are frequently used by the Distributed Objects system to represent the capabilities of an object executing on a remote system.

The syntax

@protocol Locking

- (void)lock;

- (void)unlock;

@end

denotes that there is the abstract idea of locking. By stating that the protocol is implemented in the class definition:

@interface SomeClass : SomeSuperClass <Locking>

@end

instances of SomeClass claim that they will provide an implementation for the two instance methods using whatever means they choose. Another example use of abstract specification is describing the desired behaviors of plug-ins without constraining what the implementation hierarchy should be.

Dynamic Typing

Objective-C, like Smalltalk, can use dynamic typing: an object can be sent a message that is not specified in its interface. This can allow for increased flexibility, as it allows an object to "capture" a message and send the message to a different object that can respond to the message appropriately, or likewise send the message on to another object. This behavior is known as message forwarding or delegation (see below). Alternatively, an error handler can be used in case the message cannot be forwarded. If an object does not forward a message, respond to it, or handle an error, the message is silently discarded. If messages are sent to nil (the null object pointer), they will be silently ignored or raise a generic exception, depending on compiler options.

Static typing information may also optionally be added to variables. This information is then checked at compile time. In the following four statements, increasingly specific type information is provided. The statements are equivalent at runtime, but the additional information allows the compiler to warn the programmer if the passed argument does not match the type specified.

- (void)setMyValue:(id)foo;

In the above statement, foo may be of any class.

- (void)setMyValue:(id<aProtocol>)foo;

In the above statement, foo may still be an instance of any class, but the class must conform to the aProtocol protocol.

- (void)setMyValue:(NSNumber *)foo;

In the above statement, foo must be an instance of the NSNumber class.

- (void)setMyValue:(NSNumber<aProtocol> *)foo;

In the above statement, foo must be an instance of the NSNumber class, and it must conform to the aProtocol protocol.

Dynamic typing can be a powerful feature. When implementing container classes using statically-typed languages without generics (like Java prior to version 5), the programmer is forced to write a container class for a generic type of object, and then cast back and forth between the abstract generic type and the real type. Casting, however, breaks the discipline of static typing. For instance, putting in an integer and reading out a string will produce a runtime error. This problem is addressed in, for example, Java 5 and C# with generic programming, but then container classes must be homogeneous in type. This need not be the case with dynamic typing.

Forwarding

Objective-C permits the sending of a message to an object that may not respond. Rather than responding or simply dropping the message, an object can forward the message to an object that can respond. Forwarding can be used to simplify implementation of certain design patterns, such as the Observer pattern or the Proxy pattern.

The Objective-C runtime specifies a pair of methods in Object

forwarding methods:

- (retval_t)forward:(SEL)sel args:(arglist_t)args; // with GCC

- (id)forward:(SEL)sel args:(marg_list)args; // with NeXT/Apple systems

action methods:

- (retval_t)performv:(SEL)sel args:(arglist_t)args; // with GCC

(id)performv:(SEL)sel args:(marg_list)args; // with NeXT/Apple systems

An object wishing to implement forwarding needs only to override the forwarding method with a new method to define the forwarding behavior. The action method performv:: need not be overridden, as this method merely performs an action based on the selector and arguments. Notice the SEL type, which is the type of messages in Objective-C.

Categories

During the design of Objective-C, one of the main concerns was the maintainability of large code bases. Experience from the structured programming world had shown that one of the main ways to improve code was to break it down into smaller pieces. Objective-C borrowed and extended the concept of categories from Smalltalk implementations to help with this process.

A category collects method implementations into separate files. The programmer can place groups of related methods into a category to make them more readable. For instance, one could create a "SpellChecking" category in the String object, collecting all of the methods related to spell checking into a single place.

Furthermore, the methods within a category are added to a class at run-time. Thus, categories permit the programmer to add methods to an existing class without the need to recompile that class or even have access to its source code. For example, if a system does not contain a spell checker in its String implementation, it could be added without modifying the String source code.

Methods within categories become indistinguishable from the methods in a class when the program is run. A category has full access to all of the instance variables within the class, including private variables.

If a category declares a method with the same method signature as an existing method in a class, the category’s method is adopted. Thus categories can not only add methods to a class, but also replace existing methods. This feature can

be used to fix bugs in other classes by rewriting their methods, or to cause a global change to a class's behavior within a program. If two categories have methods with the same name (not to be confused with method signature), it is undefined which category's method is adopted.

Other languages have attempted to add this feature in a variety of ways. TOM took the Objective-C system a step further and allowed for the addition of variables as well. Other languages have used prototype oriented solutions instead, with the most notable being Self.

The C# and Visual Basic.NET languages implement superficially similar functionality in the form of extension methods, but these do not have access to the private variables of the class. Ruby and several other dynamic programming languages refer to the technique as "monkey patching".

#import

In the C language, the #include pre-compile directive always causes a file's contents to be inserted into the source at that point. Objective-C has the equivalent #import directive except each file is included only once per compilation unit, obviating the need for include guards.

Objective-C 2.0

At the 2006 Worldwide Developers Conference, Apple announced the release of "Objective-C 2.0," a revision of the Objective-C language to include "modern garbage collection, syntax enhancements, runtime performance improvements, and 64-bit support". Mac OS X v10.5, released in October 2007, included an Objective-C 2.0 compiler. GCC 4.6 supports many new Objective-C features, such as declared and synthesized properties, dot syntax, fast enumeration, optional protocol methods, method/protocol/class attributes, class extensions and a new GNUnn Objective-C runtime API.

Garbage Collection

Objective-C 2.0 provides an optional conservative, yet generational garbage collector. When run in backwards-compatible mode, the runtime turns reference counting operations such as "retain" and "release" into no-ops. All objects are subject to garbage collection when garbage collection is enabled. Regular C pointers may be qualified with "__strong" to also trigger the underlying write-barrier compiler intercepts and thus participate in garbage collection.[A zero-ing weak subsystem is also provided such that pointers marked as "__weak" are set to zero when the object (or more simply, GC memory) is collected. The garbage collector does not exist on the iOS implementation of Objective-C 2.0. Garbage Collection in Objective-C runs on a low-priority background thread, and can halt on user events, with the intention of keeping the user experience responsive.

Properties

Objective-C 2.0 introduces a new syntax to declare instance variables as properties, with optional attributes to configure the generation of accessor methods. Properties are, in a sense, public instance variables; that is, declaring an instance variable as a property provides external classes with access (possibly limited, e.g. read only) to that property. A property may be declared as "readonly", and may be provided with storage semantics such as "assign", "copy" or "retain". By default, properties are considered atomic, which results in a lock preventing multiple threads from accessing them at the same time. A property can be declared as "nonatomic", which removes this lock.

```
@interface Person : NSObject {
@public
NSString *name;
@private
int age;
}
 @property(copy) NSString *name;
@property(readonly) int age;
-(id)initWithAge:(int)age;
@end
```

Properties are implemented by way of the @synthesize keyword, which generates getter (and setter, if not read-only) methods according to the property declaration. Alternatively, the getter and setter methods must be implemented explicitly, or the @dynamic keyword can be used to indicate that accessor methods will be provided by other means.

```
@implementation Person
@synthesize name;
 -(id)initWithAge:(int)initAge {
age = initAge; // NOTE: direct instance variable assignment, not property setter
return self;
}
-(int)age {
return 29;
}
@end
```

Properties can be accessed using the traditional message passing syntax, dot notation, or, in Key-Value Coding, by name via the "valueForKey:"/"setValue:forKey:" methods.

Person *aPerson = [[Person alloc] initWithAge: 53];

aPerson.name = @"Steve"; // NOTE: dot notation, uses synthesized setter, equivalent to [aPerson setName: @"Steve"];

NSLog(@"Access by message (%@), dot notation(%@), property name(%@) and direct instance variable access (%@)",

[aPerson name], aPerson.name, [aPerson valueForKey:@"name"], aPerson->name);

In order to use dot notation to invoke property accessors within an instance method, the "self" keyword should be used:

-(void) introduceMyselfWithProperties:(BOOL)useGetter {

NSLog(@"Hi, my name is %@.", (useGetter ? self.name : name)); // NOTE: getter vs. ivar access

}

A class or protocol's properties may be dynamically introspected.

int i;

int propertyCount = 0;

objc_property_t *propertyList = class_copyPropertyList([aPerson class], &propertyCount);

for (i = 0; i < propertyCount; i++) {

objc_property_t *thisProperty = propertyList + i;

const char* propertyName = property_getName(*thisProperty);

NSLog(@"Person has a property: '%s'", propertyName);

}

Non-fragile Instance Variables

Objective-C 2.0 provides non-fragile instance variables where supported by the runtime (i.e. when building 64-bit Mac OS X code as well as all iOS code). Under the modern runtime, an extra layer of indirection is added to instance variable access, allowing the dynamic linker to adjust instance layout at runtime. This feature allows for two important improvements to Objective-C code:

This eliminates the fragile binary interface problem - Superclasses can change sizes without affecting binary compatibility.

This allows instance variables that provide the backing for properties to be synthesized at runtime without them being declared in the class' interface.

Fast Enumeration

Instead of using an NSEnumerator object or indices to iterate through a collection, Objective-C 2.0 offers the fast enumeration syntax. In Objective-C 2.0, the following loops are functionally equivalent, but have different performance characteristics.

```
// Using NSEnumerator
NSEnumerator *enumerator = [thePeople objectEnumerator];
Person *p;
while ((p = [enumerator nextObject]) != nil) {
NSLog(@"%@ is %i years old.", [p name], [p age]);
}
// Using indexes
for (int i = 0; i < [thePeople count]; i++) {
Person *p = [thePeople objectAtIndex:i];
NSLog(@"%@ is %i years old.", [p name], [p age]);
}
// Using fast enumeration
for (Person *p in thePeople) {
NSLog(@"%@ is %i years old.", [p name], [p age]);
}
```

Fast enumeration generates more efficient code than standard enumeration because method calls to enumerate over objects are replaced by pointer arithmetic using the NSFastEnumeration protocol.

Library Use

Objective-C today is often used in tandem with a fixed library of standard objects (often known as a "kit" or "framework"), such as Cocoa or GNUstep. These libraries often come with the operating system: the GNUstep libraries often come with GNU/Linux based distributions and Cocoa comes with Mac OS X. The programmer is not forced to inherit functionality from the existing base class (NSObject). Objective-C allows for the declaration of new root classes that do not inherit any existing functionality. Originally, Objective-C based programming environments typically offered an Object class as the base class from which almost all other classes inherited. With the introduction of OpenStep, NeXT created a new base class named NSObject, which offered additional features over Object (an emphasis on using object references and

reference counting instead of raw pointers, for example). Almost all classes in Cocoa inherit from NSObject.

Not only did the renaming serve to differentiate the new default behavior of classes within the OpenStep API, but it allowed code that used Object—the original base class used on NeXTSTEP (and, more or less, other Objective-C class libraries)—to co-exist in the same runtime with code that used NSObject (with some limitations). The introduction of the two letter prefix also became a simplistic form of namespaces, which Objective-C lacks. Using a prefix to create an informal packaging identifier became an informal coding standard in the Objective-C community, and continues to this day.

Analysis of the Language

Objective-C implementations use a thin runtime system written in C, which adds little to the size of the application. In contrast, most object-oriented systems at the time that it was created used large virtual machine runtimes. Programs written in Objective-C tend to be not much larger than the size of their code and that of the libraries (which generally do not need to be included in the software distribution), in contrast to Smalltalk systems where a large amount of memory was used just to open a window. Objective-C applications tend to be larger than similar C or C++ applications because Objective-C dynamic typing does not allow methods to be stripped or inlined. Since the programmer has such freedom to delegate, forward calls, build selectors on the fly and pass them to the runtime system, the Objective-C compiler cannot assume it's safe to remove unused methods or to inline calls.

Likewise, the language can be implemented on top of existing C compilers (in GCC, first as a preprocessor, then as a module) rather than as a new compiler. This allows Objective-C to leverage the huge existing collection of C code, libraries, tools, etc. Existing C libraries can be wrapped in Objective-C wrappers to provide an OO-style interface. In this aspect, it is similar to GObject library and Vala language, which are widely used in development of GTK applications.

All of these practical changes lowered the barrier to entry, likely the biggest problem for the widespread acceptance of Smalltalk in the 1980s.

The first versions of Objective-C did not support garbage collection. At the time this decision was a matter of some debate, and many people considered long "dead times" (when Smalltalk did collection) to render the entire system unusable. Some 3rd party implementations have added this feature (most notably GNUstep) and Apple has implemented it as of Mac OS X v10.5.

Another common criticism is that Objective-C does not have language support for namespaces. Instead, programmers are forced to add prefixes to their class names, which are traditionally shorter than namespace names and thus more prone to collisions. As of 2007, all Mac OS X classes and functions in the Cocoa programming environment are prefixed with "NS" (e.g. NSObject, NSButton) to identify them as belonging to the Mac OS X or iOS core; the

"NS" derives from the names of the classes as defined during the development of NeXTstep.

Since Objective-C is a strict superset of C, it does not treat C primitive types as first-class objects.

Because Objective-C uses dynamic runtime typing and because all method calls are function calls (or, in some cases, syscalls), many common performance optimizations cannot be applied to Objective-C methods (for example: inlining, constant propagation, interprocedural optimizations, and scalar replacement of aggregates). This limits the performance of Objective-C abstractions relative to similar abstractions in languages such as C++ where such optimizations are possible.

Lisp

“Do the arithmetic or be doomed to talk nonsense.” John McCarthy

Lisp (or LISP) is a family of computer programming languages with a long history and a distinctive, fully parenthesized syntax. Originally specified in 1958, Lisp is the second-oldest high-level programming language in widespread use today; only Fortran is older (by one year). Like Fortran, Lisp has changed a great deal since its early days, and a number of dialects have existed over its history. Today, the most widely known general-purpose Lisp dialects are Common Lisp, Scheme, and Clojure.

Lisp was originally created as a practical mathematical notation for computer programs, influenced by the notation of Alonzo Church's lambda calculus. It quickly became the favored programming language for artificial intelligence (AI) research. As one of the earliest programming languages, Lisp pioneered many ideas in computer science, including tree data structures, automatic storage management, dynamic typing, and the self-hosting compiler.

The name LISP derives from "LISt Processing". Linked lists are one of Lisp languages' major data structures, and Lisp source code is itself made up of lists. As a result, Lisp programs can manipulate source code as a data structure, giving rise to the macro systems that allow programmers to create new syntax or even new domain-specific languages embedded in Lisp.

The interchangeability of code and data also gives Lisp its instantly recognizable syntax. All program code is written as s-expressions, or parenthesized lists. A function call or syntactic form is written as a list with the function or operator's name first, and the arguments following; for instance, a function f that takes three arguments might be called using (f arg1 arg2 arg3).

Lisp was invented by John McCarthy in 1958 while he was at the Massachusetts Institute of Technology (MIT). McCarthy published its design in a paper in Communications of the ACM in 1960, entitled "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I" ("Part II" was never published). He showed that with a few simple operators and a notation for functions, one can build a Turing-complete language for algorithms.

Information Processing Language was the first AI language, from 1955 or 1956, and already included many of the concepts, such as list-processing and recursion, which came to be used in Lisp.

McCarthy's original notation used bracketed "M-expressions" that would be translated into S-expressions. As an example, the M-expression car[cons[A,B]] is equivalent to the S-expression (car (cons A B)). Once Lisp was implemented, programmers rapidly chose to use S-expressions, and M-

expressions were abandoned. M-expressions surfaced again with short-lived attempts of MLISP by Horace Enea and CGOL by Vaughan Pratt.

Lisp was first implemented by Steve Russell on an IBM 704 computer. Russell had read McCarthy's paper, and realized (to McCarthy's surprise) that the Lisp eval function could be implemented in machine code. The result was a working Lisp interpreter which could be used to run Lisp programs, or more properly, 'evaluate Lisp expressions.'

Two assembly language macros for the IBM 704 became the primitive operations for decomposing lists: car (Contents of the Address part of Register number) and cdr (Contents of the Decrement part of Register number). From the context, it is clear that the term "Register" is used here to mean "Memory Register", nowadays called "Memory Location". Lisp dialects still use car and cdr (// kɑr/ɐand // kkdər/) for the operations that return the first item in a list and the rest of the list respectively.

The first complete Lisp compiler, written in Lisp, was implemented in 1962 by Tim Hart and Mike Levin at MIT. This compiler introduced the Lisp model of incremental compilation, in which compiled and interpreted functions can intermix freely. The language used in Hart and Levin's memo is much closer to modern Lisp style than McCarthy's earlier code.

Lisp was a difficult system to implement with the compiler techniques and stock hardware of the 1970s. Garbage collection routines, developed by then-MIT graduate student Daniel Edwards, made it practical to run Lisp on general-purpose computing systems, but efficiency was still a problem. This led to the creation of Lisp machines: dedicated hardware for running Lisp environments and programs. Advances in both computer hardware and compiler technology soon made Lisp machines obsolete.

During the 1980s and 1990s, a great effort was made to unify the work on new Lisp dialects (mostly successors to Maclisp like ZetaLisp and NIL (New Implementation of Lisp)) into a single language. The new language, Common Lisp, was somewhat compatible with the dialects it replaced (the book Common Lisp the Language notes the compatibility of various constructs). In 1994, ANSI published the Common Lisp standard, "ANSI X3.226-1994 Information Technology Programming Language Common Lisp."

Connection to Artificial Intelligence

Since its inception, Lisp was closely connected with the artificial intelligence research community, especially on PDP-10 systems. Lisp was used as the implementation of the programming language Micro Planner which was used in the famous AI system SHRDLU. In the 1970s, as AI research spawned commercial offshoots, the performance of existing Lisp systems became a growing issue.

Lisp was the first homoiconic programming language: the primary representation of program code is the same type of list structure that is also

used for the main data structures. As a result, Lisp functions can be manipulated, altered or even created within a Lisp program without extensive parsing or manipulation of binary machine code. This is generally considered one of the primary advantages of the language with regard to its expressive power, and makes the language amenable to metacircular evaluation.

The ubiquitous if-then-else structure, now taken for granted as an essential element of any programming language, was invented by McCarthy for use in Lisp, where it saw its first appearance in a more general form (the cond structure). It was inherited by ALGOL, which popularized it.

Lisp deeply influenced Alan Kay, the leader of the research on Smalltalk, and then in turn Lisp was influenced by Smalltalk, by adopting object-oriented programming features (classes, instances, etc.) in the late 1970s.

Largely because of its resource requirements with respect to early computing hardware (including early microprocessors), Lisp did not become as popular outside of the AI community as Fortran and the ALGOL-descended C language. Newer languages such as Java and Python have incorporated some limited versions of some of the features of Lisp, but are necessarily unable to bring the coherence and synergy of the full concepts found in Lisp. Because of its suitability to ill-defined, complex, and dynamic applications, Lisp is currently enjoying some resurgence of popular interest.

Symbolic expressions

Lisp is an expression-oriented language. Unlike most other languages, no distinction is made between "expressions" and "statements"; all code and data are written as expressions. When an expression is evaluated, it produces a value (in Common Lisp, possibly multiple values), which then can be embedded into other expressions. Each value can be any data type.

McCarthy's 1958 paper introduced two types of syntax: S-expressions (Symbolic expressions, also called "sexps"), which mirror the internal representation of code and data; and M-expressions (Meta Expressions), which express functions of S-expressions. M-expressions never found favor, and almost all Lisps today use S-expressions to manipulate both code and data.

The use of parentheses is Lisp's most immediately obvious difference from other programming language families. As a result, students have long given Lisp nicknames such as Lost In Stupid Parentheses, or Lots of Irritating Superfluous Parentheses. However, the S-expression syntax is also responsible for much of Lisp's power: the syntax is extremely regular, which facilitates manipulation by computer. However, the syntax of Lisp is not limited to traditional parentheses notation. It can be extended to include alternative notations. XMLisp, for instance, is a Common Lisp extension that employs the metaobject-protocol to integrate S-expressions with the Extensible Markup Language (XML).

The reliance on expressions gives the language great flexibility. Because Lisp functions are themselves written as lists, they can be processed exactly like data. This allows easy writing of programs which manipulate other programs (metaprogramming). Many Lisp dialects exploit this feature using macro systems, which enables extension of the language almost without limit.

Lists

A Lisp list is written with its elements separated by whitespace, and surrounded by parentheses. For example, (1 2 foo) is a list whose elements are three atoms: the values 1, 2, and foo. These values are implicitly typed: they are respectively two integers and a Lisp-specific data type called a "symbol", and do not have to be declared as such.

The empty list () is also represented as the special atom nil. This is the only entity in Lisp which is both an atom and a list.

Expressions are written as lists, using prefix notation. The first element in the list is the name of a form, i.e., a function, operator, macro, or "special operator" (see below.) The remainder of the list are the arguments. For example, the function list returns its arguments as a list, so the expression

(list '1 '2 'foo)

evaluates to the list (1 2 foo). The "quote" before the arguments in the preceding example is a "special operator" which prevents the quoted arguments from being evaluated (not strictly necessary for the numbers, since 1 evaluates to 1, etc.). Any unquoted expressions are recursively evaluated before the enclosing expression is evaluated. For example,

(list 1 2 (list 3 4))

evaluates to the list (1 2 (3 4)). Note that the third argument is a list; lists can be nested.

Operators

Arithmetic operators are treated similarly. The expression

(+ 1 2 3 4)

evaluates to 10. The equivalent under infix notation would be "1 + 2 + 3 + 4". Arithmetic operators in Lisp are variadic (or n-ary), able to take any number of arguments.

"Special operators" (sometimes called "special forms") provide Lisp's control structure. For example, the special operator if takes three arguments. If the first argument is non-nil, it evaluates to the second argument; otherwise, it evaluates to the third argument. Thus, the expression

(if nil

(list 1 2 "foo")

(list 3 4 "bar"))

evaluates to (3 4 "bar"). Of course, this would be more useful if a non-trivial expression had been substituted in place of nil.

Lambda Expressions

Another special operator, lambda, is used to bind variables to values which are then evaluated within an expression. This operator is also used to create functions: the arguments to lambda are a list of arguments, and the expression or expressions to which the function evaluates (the returned value is the value of the last expression that is evaluated). The expression

(lambda (arg) (+ arg 1))

evaluates to a function that, when applied, takes one argument, binds it to arg and returns the number one greater than that argument. Lambda expressions are treated no differently from named functions; they are invoked the same way. Therefore, the expression

((lambda (arg) (+ arg 1)) 5)

evaluates to 6.

Atoms

In the original LISP there were two fundamental data types: atoms and lists. A list was a finite ordered sequence of elements, where each element is in itself either an atom or a list, and an atom was a number or a symbol. A symbol was essentially a unique named item, written as an Alphanumeric string in source code, and used either as a variable name or as a data item in symbolic processing. For example, the list (FOO (BAR 1) 2) contains three elements: the symbol FOO, the list (BAR 1), and the number 2.

The essential difference between atoms and lists was that atoms were immutable and unique. Two atoms that appeared in different places in source code but were written in exactly the same way represented the same object, whereas each list was a separate object that could be altered independently of other lists and could be distinguished from other lists by comparison operators.

As more data types were introduced in later Lisp dialects, and programming styles evolved, the concept of an atom lost importance. Many dialects still retained the predicate atom for legacy compatibility, defining it true for any object which is not a cons.

Conses and Lists

A Lisp list is a singly linked list. Each cell of this list is called a cons (in Scheme, a pair), and is composed of two pointers, called the car and cdr. These are equivalent to the data and next fields discussed in the article linked list, respectively.

Of the many data structures that can be built out of cons cells, one of the most basic is called a proper list. A proper list is either the special nil (empty list)

symbol, or a cons in which the car points to a datum (which may be another cons structure, such as a list), and the cdr points to another proper list.

If a given cons is taken to be the head of a linked list, then its car points to the first element of the list, and its cdr points to the rest of the list. For this reason, the car and cdr functions are also called first and rest when referring to conses which are part of a linked list (rather than, say, a tree).

Thus, a Lisp list is not an atomic object, as an instance of a container class in C++ or Java would be. A list is nothing more than an aggregate of linked conses. A variable which refers to a given list is simply a pointer to the first cons in the list. Traversal of a list can be done by "cdring down" the list; that is, taking successive cdrs to visit each cons of the list; or by using any of a number of higher-order functions to map a function over a list.

Because conses and lists are so universal in Lisp systems, it is a common misconception that they are Lisp's only data structures. In fact, all but the most simplistic Lisps have other data structures – such as vectors (arrays), hash tables, structures, and so forth.

S-expressions Represent Lists

Parenthesized S-expressions represent linked list structures. There are several ways to represent the same list as an S-expression. A cons can be written in dotted-pair notation as (a . b), where a is the car and b the cdr. A longer proper list might be written (a . (b . (c . (d . nil)))) in dotted-pair notation. This is conventionally abbreviated as (a b c d) in list notation. An improper list may be written in a combination of the two – as (a b c . d) for the list of three conses whose last cdr is d (i.e., the list (a . (b . (c . d))) in fully specified form).

List-Processing Procedures

Lisp provides many built-in procedures for accessing and controlling lists. Lists can be created directly with the list procedure, which takes any number of arguments, and returns the list of these arguments.

```
(list 1 2 'a 3)

;Output: (1 2 a 3)

(list 1 '(2 3) 4)

;Output: (1 (2 3) 4)
```

Because of the way that lists are constructed from cons pairs, the [[cons]] procedure can be used to add an element to the front of a list. Note that the cons procedure is asymmetric in how it handles list arguments, because of how lists are constructed.

```
(cons 1 '(2 3))

;Output: (1 2 3)

(cons '(1 2) '(3 4))
```

;Output: ((1 2) 3 4)

The [[append]] procedure appends two (or more) lists to one another. Because Lisp lists are linked lists, appending two lists has asymptotic time complexity O(n)

(append '(1 2) '(3 4))

;Output: (1 2 3 4)

(append '(1 2 3) '() '(a) '(5 6))

;Output: (1 2 3 a 5 6)

Shared Structure

Lisp lists, being simple linked lists, can share structure with one another. That is to say, two lists can have the same tail, or final sequence of conses. For instance, after the execution of the following Common Lisp code:

(setf foo (list 'a 'b 'c))

(setf bar (cons 'x (cdr foo)))

the lists foo and bar are (a b c) and (x b c) respectively. However, the tail (b c) is the same structure in both lists. It is not a copy; the cons cells pointing to b and c are in the same memory locations for both lists.

Sharing structure rather than copying can give a dramatic performance improvement. However, this technique can interact in undesired ways with functions that alter lists passed to them as arguments. Altering one list, such as by replacing the c with a goose, will affect the other:

(setf (third foo) 'goose)

This changes foo to (a b goose), but thereby also changes bar to (x b goose) – a possibly unexpected result. This can be a source of bugs, and functions which alter their arguments are documented as destructive for this very reason.

Aficionados of functional programming avoid destructive functions. In the Scheme dialect, which favors the functional style, the names of destructive functions are marked with a cautionary exclamation point, or "bang"—such as set-car! (read set car bang), which replaces the car of a cons. In the Common Lisp dialect, destructive functions are commonplace; the equivalent of set-car! is named rplaca for "replace car." This function is rarely seen however as Common Lisp includes a special facility, setf, to make it easier to define and use destructive functions. A frequent style in Common Lisp is to write code functionally (without destructive calls) when prototyping, then to add destructive calls as an optimization where it is safe to do so.

Self-evaluating Forms and Quoting

Lisp evaluates expressions which are entered by the user. Symbols and lists evaluate to some other (usually, simpler) expression – for instance, a symbol

evaluates to the value of the variable it names; (+ 2 3) evaluates to 5. However, most other forms evaluate to themselves: if you enter 5 into Lisp, it returns 5.

Any expression can also be marked to prevent it from being evaluated (as is necessary for symbols and lists). This is the role of the quote special operator, or its abbreviation ' (a single quotation mark). For instance, usually if you enter the symbol foo you will get back the value of the corresponding variable (or an error, if there is no such variable). If you wish to refer to the literal symbol, you enter (quote foo) or, usually, 'foo.

Both Common Lisp and Scheme also support the backquote operator (known as quasiquote in Scheme), entered with the ` character. This is almost the same as the plain quote, except it allows expressions to be evaluated and their values interpolated into a quoted list with the comma and comma-at operators. If the variable snue has the value (bar baz) then `(foo ,snue) evaluates to (foo (bar baz)), while `(foo ,@snue) evaluates to (foo bar baz). The backquote is most frequently used in defining macro expansions.

Self-evaluating forms and quoted forms are Lisp's equivalent of literals. It may be possible to modify the values of (mutable) literals in program code. For instance, if a function returns a quoted form, and the code that calls the function modifies the form, this may alter the behavior of the function on subsequent iterations.

```
(defun should-be-constant ()
'(one two three))
 (let ((stuff (should-be-constant)))
  (setf (third stuff) 'bizarre))              ; bad!
 (should-be-constant)  ; returns (one two bizarre)
```

Modifying a quoted form like this is generally considered bad style, and is defined by ANSI Common Lisp as erroneous (resulting in "undefined" behavior in compiled files, because the file-compiler can coalesce similar constants, put them in write-protected memory, etc.).

Lisp's formalization of quotation has been noted by Douglas Hofstadter (in Gödel, Escher, Bach) and others as an example of the philosophical idea of self-reference.

Scope and Closure

The modern Lisp family splits over the use of dynamic or static (aka lexical) scope. Clojure, Common Lisp and Scheme make use of static scoping by default, while Newlisp, Picolisp and the embedded languages in Emacs and AutoCAD use dynamic scoping.

List Structure of Program Code

A fundamental distinction between Lisp and other languages is that in Lisp, the textual representation of a program is simply a human-readable description of

the same internal data structures (linked lists, symbols, number, characters, etc.) as would be used by the underlying Lisp system.

Lisp macros operate on these structures. Because Lisp code has the same structure as lists, macros can be built with any of the list-processing functions in the language. In short, anything that Lisp can do to a data structure, Lisp macros can do to code. In contrast, in most other languages, the parser's output is purely internal to the language implementation and cannot be manipulated by the programmer. Macros in C, for instance, operate on the level of the preprocessor, before the parser is invoked, and cannot re-structure the program code in the way Lisp macros can.

In simplistic Lisp implementations, this list structure is directly interpreted to run the program; a function is literally a piece of list structure which is traversed by the interpreter in executing it. However, most actual Lisp systems (including all conforming Common Lisp systems) also include a compiler. The compiler translates list structure into machine code or bytecode for execution.

Evaluation and the Read–Eval–Print Loop

Lisp languages are frequently used with an interactive command line, which may be combined with an integrated development environment. The user types in expressions at the command line, or directs the IDE to transmit them to the Lisp system. Lisp reads the entered expressions, evaluates them, and prints the result. For this reason, the Lisp command line is called a "read–eval–print loop", or REPL.

The basic operation of the REPL is as follows. This is a simplistic description which omits many elements of a real Lisp, such as quoting and macros.

The read function accepts textual S-expressions as input, and parses them into an internal data structure. For instance, if you type the text (+ 1 2) at the prompt, read translates this into a linked list with three elements: the symbol +, the number 1, and the number 2. It so happens that this list is also a valid piece of Lisp code; that is, it can be evaluated. This is because the car of the list names a function—the addition operation.

Note that a foo will be read as a single symbol. 123 will be read as the number 123. "123" will be read as the string "123".

The eval function evaluates the data, returning zero or more other Lisp data as a result. Evaluation does not have to mean interpretation; some Lisp systems compile every expression to native machine code. It is simple, however, to describe evaluation as interpretation: To evaluate a list whose car names a function, eval first evaluates each of the arguments given in its cdr, then applies the function to the arguments. In this case, the function is addition, and applying it to the argument list (1 2) yields the answer 3. This is the result of the evaluation.

The symbol foo evaluates to the value of the symbol foo. Data like the string "123" evaluates to the same string. The list (quote (1 2 3)) evaluates to the list (1 2 3).

It is the job of the print function to represent output to the user. For a simple result such as 3 this is trivial. An expression which evaluated to a piece of list structure would require that print traverse the list and print it out as an S-expression.

To implement a Lisp REPL, it is necessary only to implement these three functions and an infinite-loop function. (Naturally, the implementation of eval will be complicated, since it must also implement all special operators like if or lambda.) This done, a basic REPL itself is but a single line of code: (loop (print (eval (read)))).

The Lisp REPL typically also provides input editing, an input history, error handling and an interface to the debugger.

Lisp is usually evaluated eagerly. In Common Lisp, arguments are evaluated in applicative order ('leftmost innermost'), while in Scheme order of arguments is undefined, leaving room for optimization by a compiler.

Control Structures

Lisp originally had very few control structures, but many more were added during the language's evolution. (Lisp's original conditional operator, cond, is the precursor to later if-then-else structures.)

Programmers in the Scheme dialect often express loops using tail recursion. Scheme's commonality in academic computer science has led some students to believe that tail recursion is the only, or the most common, way to write iterations in Lisp, but this is incorrect. All frequently seen Lisp dialects have imperative-style iteration constructs, from Scheme's do loop to Common Lisp's complex loop expressions. Moreover, the key issue that makes this an objective rather than subjective matter is that Scheme makes specific requirements for the handling of tail calls, and consequently the reason that the use of tail recursion is generally encouraged for Scheme is that the practice is expressly supported by the language definition itself. By contrast, ANSI Common Lisp does not require the optimization commonly referred to as tail call elimination. Consequently, the fact that tail recursive style as a casual replacement for the use of more traditional iteration constructs (such as do, dolist or loop) is discouraged in Common Lisp is not just a matter of stylistic preference, but potentially one of efficiency (since an apparent tail call in Common Lisp may not compile as a simple jump) and program correctness (since tail recursion may increase stack use in Common Lisp, risking stack overflow).

Some Lisp control structures are special operators, equivalent to other languages' syntactic keywords. Expressions using these operators have the same surface appearance as function calls, but differ in that the arguments are

not necessarily evaluated—or, in the case of an iteration expression, may be evaluated more than once.

In contrast to most other major programming languages, Lisp allows the programmer to implement control structures using the language itself. Several control structures are implemented as Lisp macros, and can even be macro-expanded by the programmer who wants to know how they work.

Both Common Lisp and Scheme have operators for non-local control flow. The differences in these operators are some of the deepest differences between the two dialects. Scheme supports re-entrant continuations using the call/cc procedure, which allows a program to save (and later restore) a particular place in execution. Common Lisp does not support re-entrant continuations, but does support several ways of handling escape continuations.

Frequently, the same algorithm can be expressed in Lisp in either an imperative or a functional style. As noted above, Scheme tends to favor the functional style, using tail recursion and continuations to express control flow. However, imperative style is still quite possible. The style preferred by many Common Lisp programmers may seem more familiar to programmers used to structured languages such as C, while that preferred by Schemers more closely resembles pure-functional languages such as Haskell.

Because of Lisp's early heritage in list processing, it has a wide array of higher-order functions relating to iteration over sequences. In many cases where an explicit loop would be needed in other languages (like a for loop in C) in Lisp the same task can be accomplished with a higher-order function. (The same is true of many functional programming languages.)

A good example is a function which in Scheme is called map and in Common Lisp is called mapcar. Given a function and one or more lists, mapcar applies the function successively to the lists' elements in order, collecting the results in a new list:

(mapcar #'+ '(1 2 3 4 5) '(10 20 30 40 50))

This applies the + function to each corresponding pair of list elements, yielding the result (11 22 33 44 55).

Examples

Here are examples of Common Lisp code.

The basic "Hello world" program:

(print "Hello world")

As the reader may have noticed from the above discussion, Lisp syntax lends itself naturally to recursion. Mathematical problems such as the enumeration of recursively defined sets are simple to express in this notation.

Evaluate a number's factorial:

(defun factorial (n)

```
(if (<= n 1)
1
(* n (factorial (- n 1)))))
```

An alternative implementation, often faster than the previous version if the Lisp system has tail recursion optimization:

```
(defun factorial (n &optional (acc 1))
(if (<= n 1)
acc
(factorial (- n 1) (* acc n))))
```

Contrast with an iterative version which uses Common Lisp's loop macro:

```
(defun factorial (n)
(loop for i from 1 to n
for fac = 1 then (* fac i)
finally (return fac)))
```

The following function reverses a list. (Lisp's built-in reverse function does the same thing.)

```
(defun -reverse (list)
(let ((return-value '()))
(dolist (e list) (push e return-value))
return-value))
```

Object Systems

Various object systems and models have been built on top of, alongside, or into Lisp, including:

The Common Lisp Object System, CLOS, is an integral part of ANSI Common Lisp. CLOS descended from New Flavors and CommonLOOPS. ANSI Common Lisp was the first standardized object-oriented programming language (1994, ANSI X3J13).

ObjectLisp or Object Lisp, used by Lisp Machines Incorporated and early versions of Macintosh Common Lisp

LOOPS (Lisp Object-Oriented Programming System) and the later CommonLOOPS

Flavors, built at MIT, and its descendant New Flavors (developed by Symbolics).

KR (short for Knowledge Representation), a constraints-based object system developed to aid the writing of Garnet, a GUI library for Common Lisp.

KEE used an object system called UNITS and integrated it with an inference engine and a truth maintenance system (ATMS).

Conceptive C

If something is better programmed in C or Objective C, that is fine Conceptive C, is for handling higher level concepts.

So in Conceptive C we have:

concept object that can change at run time

idea (id) pointer to concept

meme apply function to concept memory data

So we leave Objective C syntax as it is and use normal inheritance and classes.

The only difference is:

```
id      object;         // would define an object
idea    concept;        // would define a concept
```

Concepts will work differently to objects but can be mixed in the same program.

Using Concepts

To use a Concept is the same as using an Object in Objective-C. Concepts can be mixed with Objects and C language can be used as Conceptive-C is

```
idea concept;
[concept        meme: argument];
```

Syntax

Conceptive-C is a thin layer on top of Objective-C, and moreover is a strict superset of C; it is possible to compile any C program with an Conceptive-C compiler, and to freely include C code within an Conceptive-C class.

The syntax for implementation and classes is the same as in Objective-C.

It should be possible to use an idea in a method that uses an object by:

```
[object   method: (id) concept];  // The pointer to concept is passed to object
```

Messages

The Conceptive-C model of conceptual programming is based on message passing to concept instances. In Conceptive-C you send a message to a concept or an Object.

In some ways, I'm still not sure if there is a language, it may be Conceptual and all be written in Objective-C. By Conceptual, I mean abstract idea, not in need of implementaion. Which would make this really a book about using Objective-C for AI programming. I mean, isn't this all just Objective C?

Wouldn't it be better to just define a concept as an Object and make a library that adds new classes that deals with ideas and concepts?

Interfaces and Implementations

Conceptive-C requires that the interface and implementation of a class be in separately declared code blocks. By convention, developers place the interface in a header file and the implementation in a code file. The header files, normally suffixed .h, are similar to C header files while the implementation (method) files, normally suffixed .m, can be very similar to C code files. No change to files same as used in Objective-C.

Interface

The interface of a class is usually defined in a header file. A common convention is to name the header file after the name of the class, e.g. Ball.h would contain the interface for the class Ball.

The use of interfaces is exactly the same as in Objective-C.

Instantiation

Once an Conceptive-C class is written, it can be instantiated. This is done by allocating an uninitialized instance of the class (an concept). An concept does not need to be initialized.

Instantiation with the default, no-parameter initializer:

MyConcept *c = [MyConcept new];

Dynamic Typing

Conceptive-C, like Smalltalk, can use dynamic typing: an object can be sent a message that is not specified in its interface. This can allow for increased flexibility, as it allows an object to "capture" a message and send the message to a different object that can respond to the message appropriately, or likewise send the message on to another object.

Changes from Objective-C

Really the only real change is the addition of an Idea, which is a pointer to a Concept. Concepts are abstract Objects used for implementing thinking machines.

Maybe I said it’s a small change, but it will work together with C and Objective C.

So does adding Idea to our compiler change much? How do we implement it?

Right now, I’m not sure.

I may be doing this all wrong. As of now Concepts are an unknown. I don’t know how they are implemented or what they can do yet. This is KISS (Keep it simple stupid) I think all I really need to define extra is a Concept and an Idea, a pointer to a Concept.

The Concept an idea points to can be changed at runtime time as the Concept is updated or renewed.

That is when our program learns something new.

So how does it do that? And what exactly are we doing?

Well we are using the compiler at run-time.

When our program decides it wants to learn something it has various options.

The simple option is write data to a data file, that can be read at startup or while it is running.

Another option would be to write to a concept memory that would work similarly to the way people remember things. The main problem with that is, I haven't implemented conceptual memory yet.

Machine Intelligence

The option that I am talking about for Machine Learning is to run the compiler and compile new code to deal with the problem in the future. We can either redefine code that we have or write new code for a new problem and add it too our program.

In FORTH language a programmer always has access to the compiler even when a program is running. So I m proposing that we steal this functionality and add it to a C compiler.

We are still going to have the problem of how do we write code that writes code?

And we don't want to it to be able to write stuff that crashes our program, so we will need limits to what can be re-written. You don't give a child a loaded gun to play with, you might give him a toy gun.

New Ideas

Some of the things that I think will make it into Conceptive C.

Conceptual Memory is an abstract data storage with no type and no fixed size. It grows and shrinks, doing automatic garbage collection as needed, like a sponge.

A Cyberspace Computer or Virtual Computer is an abstract computer, not physical. It may have a defined set of registers, an address space, machine opcodes and a clock. It can be given a program to run. It will input and out put through defined streams, portals or bitmaps.

A Cell is a Conceptual Cyberspace Computer, that can copy itself and run multiple copies of itself on multiple processors. It can send messages to other cells, concepts or objects.

A Word is a Concept that understands the use of itself in a sentence.

A Sentence is a collection of words, it can be a fixed list or just a group.

An Idea is a pointer to a Concept.

A Concept is an Object that can change itself at run time.

A Mind Concept or Intelligence is a collection of Concepts used for thinking.

A Dictionary Concept is a list of Words with definitions and types.

A Wikipedia or Wiki Concept is a Knowledge Database.

A World Map is map of the World used to store Knowledge.

A Body Concept is an group of Objects controlled by a Mind.

A Concept Bot is a question and answer conversation Bot able to access a Dictionary or Wikipedia or search the Internet.

A Concept Map is a linked list of Concepts.

A Clock Concept is something that provides a time code or can trigger events.

A Frame Concept will trigger events on Fame Buffer Events.

A Conceptual Portal is used to input and output data to a Concept, the data can be Text, Object, Audio, Video or 2D Graphic Bitmaps or 3D Graphic Wire Frames or Objects.

Conceptual Memory could work as a dictionary, a Wikipedia, a World Map or a Digiverse.

Clock and Frame Concepts could be part of Portals.

Cells could swallow up Virtual Computers.

A Mind could be a Concept Bot.

A Cell has a Mind and a Body.

So we get Idea, Concept, Memory, Mind, Map, Body, Portal and Cell as the Core Concepts of Conceptive C.

Idea

An idea is a pointer to a concept. I've used the word idea as I am trying to promote the use of concepts and ideas as part of the programming language. As we are trying to program an intelligence it will help to think of ideas and concepts as the tools to let of think and understand the problem of how do we think about something.

In the same way that an id is a pointer to an object in objective-c.

Do we need to cast from an id to an idea?

id = idea;

idea = id;

or do we need to use

id = (id)idea;

idea = (idea)id;

This is an implementation issue.

it would really be

id object;

object MyObject;

idea concept;

concept MyConcept;

// This would instantiate a object but store it in an concept pointer

MyConcept = (idea)[[MyObject alloc] init];

or

// This would instantiate a concept but store it in an object pointer

MyObject = (id)[MyConcept new];

Although you shouldn't really do it the compiler should see them as equivalent pointers and not need a cast.

In the most narrow sense, an idea is just whatever is before the mind when one thinks. Very often, ideas are constructed as representational images; i.e. images of some object. In other contexts, ideas are taken to be concepts, although abstract concepts do not necessarily appear as images. Many philosophers consider ideas to be a fundamental ontological category of being.

The capacity to create and understand the meaning of ideas is considered to be an essential and defining feature of human beings. In a popular sense, an idea arises in a reflex, spontaneous manner, even without thinking or serious reflection, for example, when we talk about the idea of a person or a place.

Innate and Adventitious Ideas

One view on the nature of ideas is that there exist some ideas (called innate ideas) which are so general and abstract, that they could not have arisen as a representation of any object of our perception, but rather were, in some sense, always in the mind before we could learn them. These are distinguished from adventitious ideas which are images or concepts which are accompanied by the judgment that they are caused by some object outside of the mind.

Another view holds that we only discover ideas in the same way that we discover the real world, from personal experiences. The view that humans acquire all or almost all their behavioral traits from nurture (life experiences) is known as tabula rasa ("blank slate"). Most of the confusions in the way of ideas arise at least in part from the use of the term "idea" to cover both the representation percept and the object of conceptual thought. This can be illustrated in terms of the doctrines of innate ideas, "concrete ideas versus abstract ideas", as well as "simple ideas versus complex ideas".

Plato

Plato was one of the earliest philosophers to provide a detailed discussion of ideas. He considered the concept of idea in the realm of metaphysics and its implications for epistemology. He asserted that there is a realm of Forms or Ideas, which exist independently of anyone who may have thought of these ideas. Material things are then imperfect and transient reflections or instantiations of the perfect and unchanging ideas. From this it follows that these Ideas are the principal reality (see also idealism). In contrast to the individual objects of sense experience, which undergo constant change and flux, Plato held that ideas are perfect, eternal, and immutable. Consequently, Plato considered that knowledge of material things is not really knowledge; real knowledge can only be had of unchanging ideas.

René Descartes

Descartes often wrote of the meaning of idea as an image or representation, often but not necessarily "in the mind", which was well known in the vernacular. Despite that Descartes is usually credited with the invention of the non-Platonic use of the term, he at first followed this vernacular use.b In his Meditations on First Philosophy he says, "Some of my thoughts are like images of things, and it is to these alone that the name 'idea' properly belongs." He sometimes maintained that ideas were innate and uses of the term idea diverge from the original primary scholastic use. He provides multiple non-equivalent definitions of the term, uses it to refer to as many as six distinct kinds of entities, and divides ideas inconsistently into various genetic categories.

For him knowledge took the form of ideas and philosophical investigation is the deep consideration of these ideas. Many times however his thoughts of knowledge and ideas were like those of Plotinus and the Neoplatonists. In Neoplatonism, the Intelligence (Nous) is the true first principle—the

determinate, referential "foundation" (arkhe)—of all existents; for it is not a self-sufficient entity like the One, but rather possesses the ability or capacity to contemplate both the One, as its prior, as well as its own thoughts, which Plotinus identifies with the Platonic Ideas or Forms (eide). A non-philosophical definition of Nous is good sense (a.k.a. "common sense"). Descartes is quoted as saying, "Of all things, good sense is the most fairly distributed: everyone thinks he is so well supplied with it that even those who are the hardest to satisfy in every other respect never desire more of it than they already have."q:René Descartes

John Locke

In striking contrast to Plato's use of idea is that of John Locke. In his Introduction to An Essay Concerning Human Understanding, Locke defines idea as "that term which, I think, serves best to stand for whatsoever is the object of the understanding when a man thinks, I have used it to express whatever is meant by phantasm, notion, species, or whatever it is which the mind can be employed about in thinking; and I could not avoid frequently using it." He said he regarded the book necessary to examine our own abilities and see what objects our understandings were, or were not, fitted to deal with. In his philosophy other outstanding figures followed in his footsteps—Hume and Kant in the 18th century, Arthur Schopenhauer in the 19th century, and Bertrand Russell, Ludwig Wittgenstein, and Karl Popper in the 20th century. Locke always believed in good sense—not pushing things to extremes and on taking fully into account the plain facts of the matter. He considered his common sense ideas "good-tempered, moderate, and down-to-earth."

David Hume

Hume differs from Locke by limiting idea to the more or less vague mental reconstructions of perceptions, the perceptual process being described as an "impression." Hume shared with Locke the basic empiricist premise that it is only from life experiences (whether their own or others') that humans' knowledge of the existence of anything outside of themselves can be ultimately derived, that they shall carry on doing what they are prompted to do by their emotional drives of varying kinds. In choosing the means to those ends, they shall follow their accustomed associations of ideas.d Hume has contended and defended the notion that "reason alone is merely the 'slave of the passions'."

Immanuel Kant

Immanuel Kant defines an idea as opposed to a concept. "Regulator ideas" are ideals that one must tend towards, but by definition may not be completely realized. Liberty, according to Kant, is an idea. The autonomy of the rational and universal subject is opposed to the determinism of the empirical subject. Kant felt that it is precisely in knowing its limits that philosophy exists. The business of philosophy he thought was not to give rules, but to analyze the private judgements of good common sense.e

Rudolf Steiner

Whereas Kant declares limits to knowledge ("we can never know the thing in itself"), in his epistemological work, Rudolf Steiner sees ideas as "objects of experience" which the mind apprehends, much as the eye apprehends light. In Goethean Science (1883), he declares, "Thinking ... is no more and no less an organ of perception than the eye or ear. Just as the eye perceives colors and the ear sounds, so thinking perceives ideas." He holds this to be the premise upon which Goethe made his natural-scientific observations.

Wilhelm Wundt

Wundt widens the term from Kant's usage to include conscious representation of some object or process of the external world. In so doing, he includes not only ideas of memory and imagination, but also perceptual processes, whereas other psychologists confine the term to the first two groups. One of Wundt's main concerns was to investigate conscious processes in their own context by experiment and introspection. He regarded both of these as exact methods, interrelated in that experimentation created optimal conditions for introspection. Where the experimental method failed, he turned to other objectively valuable aids, specifically to those products of cultural communal life which lead one to infer particular mental motives. Outstanding among these are speech, myth, and social custom. Wundt designed the basic mental activity apperception—a unifying function which should be understood as an activity of the will. Many aspects of his empirical physiological psychology are used today. One is his principles of mutually enhanced contrasts and of assimilation and dissimilation (i.e. in color and form perception and his advocacy of objective methods of expression and of recording results, especially in language. Another is the principle of heterogony of ends—that multiply motivated acts lead to unintended side effects which in turn become motives for new actions.

Charles Sanders Peirce

C.S. Peirce published the first full statement of pragmatism in his important works "How to Make Our Ideas Clear" (1878) and "The Fixation of Belief" (1877). In "How to Make Our Ideas Clear" he proposed that a clear idea (in his study he uses concept and idea as synonymic) is defined as one, when it is apprehended such as it will be recognized wherever it is met, and no other will be mistaken for it. If it fails of this clearness, it is said to be obscure. He argued that to understand an idea clearly we should ask ourselves what difference its application would make to our evaluation of a proposed solution to the problem at hand. Pragmatism (a term he appropriated for use in this context), he defended, was a method for ascertaining the meaning of terms (as a theory of meaning). The originality of his ideas is in their rejection of what was accepted as a view and understanding of knowledge by scientists for some 250 years, i.e. that, he pointed, knowledge was an impersonal fact. Peirce contended that we acquire knowledge as participants, not as spectators. He felt "the real" is which, sooner or later, information acquired through ideas and

knowledge with the application of logical reasoning would finally result in. He also published many papers on logic in relation to ideas.

G.F. Stout and J.M. Baldwin

G.F. Stout and J.M. Baldwin, in the Dictionary of Philosophy and Psychology, define idea as "the reproduction with a more or less adequate image, of an object not actually present to the senses." They point out that an idea and a perception are by various authorities contrasted in various ways. "Difference in degree of intensity", "comparative absence of bodily movement on the part of the subject", "comparative dependence on mental activity", are suggested by psychologists as characteristic of an idea as compared with a perception.

It should be observed that an idea, in the narrower and generally accepted sense of a mental reproduction, is frequently composite. That is, as in the example given above of the idea of chair, a great many objects, differing materially in detail, all call a single idea. When a man, for example, has obtained an idea of chairs in general by comparison with which he can say "This is a chair, that is a stool", he has what is known as an "abstract idea" distinct from the reproduction in his mind of any particular chair (see abstraction). Furthermore a complex idea may not have any corresponding physical object, though its particular constituent elements may severally be the reproductions of actual perceptions. Thus the idea of a centaur is a complex mental picture composed of the ideas of man and horse, that of a mermaid of a woman and a fish.

Anthropology and the Social Sciences

Diffusion studies explore the spread of ideas from culture to culture. Some anthropological theories hold that all cultures imitate ideas from one or a few original cultures, the Adam of the Bible or several cultural circles that overlap. Evolutionary diffusion theory holds that cultures are influenced by one another, but that similar ideas can be developed in isolation.

In mid-20th century, social scientists began to study how and why ideas spread from one person or culture to another. Everett Rogers pioneered diffusion of innovations studies, using research to prove factors in adoption and profiles of adopters of ideas. In 1976, in his book The Selfish Gene, Richard Dawkins suggested applying biological evolutionary theories to the spread of ideas. He coined the term meme to dibe an abstract unit of selection, equivalent to the gene in evolutionary biology.

Dr. Samuel Johnson

James Boswell recorded Dr.Samuel Johnson's opinion about ideas. Johnson claimed that they are mental images or internal visual pictures. As such, they have no relation to words or the concepts which are designated by verbal names.

He was particularly indignant against the almost universal use of the word idea in the sense of notion or opinion, when it is clear that idea can only signify something of which an image can

be formed in the mind. We may have an idea or image of a mountain, a tree, a building; but we cannot surely have an idea or image of an argument or proposition. Yet we hear the sages of the law 'delivering their ideas upon the question under consideration;' and the first speakers in parliament 'entirely coinciding in the idea which has been ably stated by an honourable member;' — or 'reprobating an idea unconstitutional, and fraught with the most dangerous consequences to a great and free country.' Johnson called this 'modern cant.'

— Boswell's Life of Johnson, Tuesday, 23 September 1777

Validity of Ideas

In the objective worth of our ideas there remains the problem of the validity. As all cognition is by ideas, it is obvious that the question of the validity of our ideas in this broad sense is that of the truth of our knowledge as a whole. Otherwise to dispute this is to take up the position of skepticism. This has often been pointed out as a means of intellectual suicide. Any chain of reasoning (common sense) by which we attempt to demonstrate the falsity of our ideas has to employ the very concept of ideas itself. Then insofar as such reasoning demands assent to the conclusion, it implies belief in the validity of all the ideas employed in the premises of the argument.

To assent the fundamental mathematical and logical axioms, including that of the principle of contradiction, implies admission of the truth of the ideas expressed in these principles. With respect to the objective worth of ideas, as involved in perception generally, the question raised is that of the existence of an independent material world comprising other human beings. The idealism of David Hume and John Stuart Mill would lead logically to solipsism (the denial of any others besides ourselves). The main foundation of all idealism and skepticism is the assumption (explicit or implicit), that the mind can never know what is outside of itself. This is to say that an idea as a cognition can never go outside of itself. This can be further expressed as we can never reach to and mentally apprehend anything outside of anything of what is actually a present state of our own consciousness.

First, this is based on a prior assumption for which no real proof is or can be given

Second, it is not only not self-evident, but directly contrary to what our mind affirms to be our direct intellectual experience.

What is possible for a human mind to apprehend cannot be laid down beforehand. It must be ascertained by careful observations and by study of the process of cognition. This postulates that the mind cannot apprehend or cognize any reality existing outside of itself and is not only a self-evident proposition, it is directly contrary to what such observation and the testimony of mankind affirms to be our actual intellectual experience.

John Stuart Mill and most extreme idealists have to admit the validity of memory and expectation. This is to say that in every act of memory or expectation which refers to any experience outside the present instant, our cognition is transcending the present modifications of the mind and judging

about reality beyond and distinct from the present states of consciousness. Considering the question as specially concerned with universal concepts, only the theory of moderate realism adopted by Aristotle and Saint Thomas can claim to guarantee objective value to our ideas. According to the nominalist and conceptualist theories there is no true correlate in rerum naturâ corresponding to the universal term.

Mathematics, astronomy, physics, chemistry, and the rest claim that their universal propositions are true and deal with realities. It is involved in the very notion of science that the physical laws formulated by the mind do mirror the working of agents in the external universe. The general terms of these sciences and the ideas which they signify have objective correlatives in the common natures and essences of the objects with which these sciences deal. Otherwise these general statements are unreal and each science is nothing more than a consistently arranged system of barren propositions deduced from empty arbitrary definitions. These postulates then have no more genuine objective value than any other coherently devised scheme of artificial symbols standing for imaginary beings. However the fruitfulness of science and the constant verifications of its predictions are incompatible with such a hypothesis.

Concept

If someone is explaining a new idea to you at what point do you feel you have understood the idea? What will make you say “I’m sorry but I didn’t understand what you just said”? The person explaining it, thinks he understands the idea.

A concept is similar to an object. It has data and methods that can act on that data.

In fact at this time as there is no Conceptive C compiler, a concept is purely conceptual and can be defined using objective C as:

```
id          concept;
```

So it is as the same level as an object. Any examples at present will use that concept as the starting point. When and if I implement a Conceptive-C compiler it would change to:

```
idea        concept;
```

So the difference is once we have a Conceptive-C compiler, we should be able to add data and methods to a concept at run time.

How do we do this?

It is not done inside an editor while editing a source file and then running a compiler and linker.

It is done by the running program.

So the running program needs to have a way of adding data, allocating memory and adding computer code, while it is running.

It could be possible for the programmer to do part of this at compile time by specifying conditions that would lead to new code or data being added to a concept. Or by having concept templates and changing the concept to a new concept to handle a problem.

One example would be a bot running and examining input text and creating word concepts from the text. If we had a class or template for a noun, adverb or adjective we could create a new word using one of these classes.

The problem is say the new word is "car". A "car" is a noun word concept or an object.

Which is it better to implement it as?

The noun concept could be used to allow a bot to use it in a conversation.

The object might be better for using it to control a robot radio control car or a car in a game.

The thing is in this situation only the noun word concept is available to add the object the program would need to be re-compiled. Our program is not likely to be in the position of having to control a radio controlled car.

But it may find out information about cars from searching the Internet.

Ideally the running program should be able to store new concepts that are created while running. It should also be possible to run a program that would turn the stored new objects into source code .m and .h files, so that could be added into the programs source code.

The other option would be to add car to a word database, rather than adding a noun concept. The problem that I see is a computer program can only program code that has been foreseen or planned for by a human programmer at this time.

At some time it may be possible for a computer program to run and define it's own goals and invent computer code to help it achieve those goals. But I don't think we are near that time yet.

What is a Concept?

A concept (substantive term: conception) is a cognitive unit of meaning—an abstract idea or a mental symbol sometimes defined as a "unit of knowledge," built from other units which act as a concept's characteristics. A concept is typically associated with a corresponding representation in a language or symbology; however, some concepts do not have a linguistic representation, which can make them more difficult to understand depending on a person's native language, such as a single meaning of a term.

There are prevailing theories in contemporary philosophy which attempt to explain the nature of concepts. The representational theory of mind proposes that concepts are mental representations, while the semantic theory of concepts (originating with Frege's distinction between concept and object) holds that they are abstract objects. Ideas are taken to be concepts, although abstract concepts do not necessarily appear to the mind as images as some ideas do. Many philosophers consider concepts to be a fundamental ontological category of being.

The term "concept" is traced back to 1554–60 (Latin conceptum - "something conceived"), but what is today termed "the classical theory of concepts" is the theory of Aristotle on the definition of terms. The meaning of "concept" is explored in mainstream information science, cognitive science, metaphysics, and philosophy of mind. In computer and information science contexts, especially, the term 'concept' is often used in unclear or inconsistent ways.

Origin and Acquisition of Concepts

John Locke's description of a general idea corresponds to a description of a concept. According to Locke, a general idea is created by abstracting, drawing away, or removing the uncommon characteristic or characteristics from several particular ideas. The remaining common characteristic is that which is similar to all of the different individuals. For example, the abstract general idea or concept that is designated by the word "red" is that characteristic which is common to apples, cherries, and blood. The abstract general idea or concept that is signified by the word "dog" is the collection of those characteristics which are common to Airedales, Collies, and Chihuahuas.

In the same tradition as Locke, John Stuart Mill stated that general conceptions are formed through abstraction. A general conception is the common element among the many images of members of a class. "...[W]hen we form a set of phenomena into a class, that is, when we compare them with one another to ascertain in what they agree, some general conception is implied in this mental operation" (A System of Logic, Book IV, Ch. II). Mill did not believe that concepts exist in the mind before the act of abstraction. "It is not a law of our intellect, that, in comparing things with each other and taking note of their agreement, we merely recognize as realized in the outward world something that we already had in our minds. The conception originally found its way to us as the result of such a comparison. It was obtained (in metaphysical phrase) by abstraction from individual things".

For Schopenhauer, empirical concepts "...are mere abstractions from what is known through intuitive perception, and they have arisen from our arbitrarily thinking away or dropping of some qualities and our retention of others." (Parerga and Paralipomena, Vol. I, "Sketch of a History of the Ideal and the Real"). In his On the Will in Nature, "Physiology and Pathology," Schopenhauer said that a concept is "drawn off from previous images ... by putting off their differences. This concept is then no longer intuitively perceptible, but is denoted and fixed merely by words." Nietzsche, who was heavily influenced by Schopenhauer, wrote: "Every concept originates through our equating what is unequal. No leaf ever wholly equals another, and the concept 'leaf' is formed through an arbitrary abstraction from these individual differences, through forgetting the distinctions..."

By contrast to the above philosophers, Immanuel Kant held that the account of the concept as an abstraction of experience is only partly correct. He called those concepts that result of abstraction "a posteriori concepts" (meaning concepts that arise out of experience). An empirical or an a posteriori concept is a general representation (Vorstellung) or non-specific thought of that which is common to several specific perceived objects.

A concept is a common feature or characteristic. Kant investigated the way that empirical a posteriori concepts are created.

The logical acts of the understanding by which concepts are generated as to their form are:

Comparison, the likening of mental images to one another in relation to the unity of consciousness.

Reflection, the going back over different mental images, how they can be comprehended in one consciousness; and finally.

Abstraction or the segregation of everything else by which the mental images differ.

In order to make our mental images into concepts, one must thus be able to compare, reflect, and abstract, for these three logical operations of the understanding are essential and general conditions of generating any concept whatever. For example, I see a fir, a willow, and a linden. In firstly comparing these objects, I notice that they are different from one another in respect of trunk, branches, leaves, and the like; further, however, I reflect only on what they have in common, the trunk, the branches, the leaves themselves, and abstract from their size, shape, and so forth; thus I gain a concept of a tree.

Pure Concepts

Kant declared that human minds possess pure or a priori concepts. Instead of being abstracted from individual perceptions, like empirical concepts, they originate in the mind itself. He called these concepts categories, in the sense of the word that means predicate, attribute, characteristic, or quality. But these pure categories are predicates of things in general, not of a particular thing.

According to Kant, there are 12 categories that constitute the understanding of phenomenal objects. Each category is that one predicate which is common to multiple empirical concepts. In order to explain how an a priori concept can relate to individual phenomena, in a manner analogous to an a posteriori concept, Kant employed the technical concept of the schema.

Conceptual Structure

It seems intuitively obvious that concepts must have some kind of structure. Up until recently, the dominant view of conceptual structure was a containment model, associated with the classical view of concepts. According to this model, a concept is endowed with certain necessary and sufficient conditions in their description which unequivocally determine an extension. The containment model allows for no degrees; a thing is either in, or out, of the concept's extension. By contrast, the inferential model understands conceptual structure to be determined in a graded manner, according to the tendency of the concept to be used in certain kinds of inferences. As a result, concepts do not have a kind of structure that is in terms of necessary and sufficient conditions; all conditions are contingent.

However, some theorists claim that primitive concepts lack any structure at all. For instance, Jerry Fodor presents his Asymmetric Dependence Theory as a way of showing how a primitive concept's content is determined by a reliable relationship between the information in mental contents and the world. These

sorts of claims are referred to as "atomistic", because the primitive concept is treated as if it were a genuine atom.

One Possible Structure

Concepts are formed by people's (or other) minds while reflecting upon their environment, subject to their sensory organs/sensors and the way such minds are in contact with their immediate and distant environment. The location of concepts is therefore assumed to be within the mind of such an organism/mechanism, more specifically in their head or equivalent place deemed to be the organ used for thinking (system of nerves or equivalent). However Concepts can be expressed in language and externalized by writing or other means such as Wikipedia. Others hold different views, such as Carl Jung, who holds that concepts may be attributed to space other than within the inside boundaries of any body or mass or material formation of living creatures. In fact some people even assume that inanimate object also have such property as "a concept" within their own solid structure.

The Dual Nature of Concepts

Clearly, the location of concepts is not decided for good yet, but it looks certain that they are related to the external world or the environment, of which of course such a living and "thinking" creature is a part of. Thus a concept is started from outside, in the relation of conception, hence the subject is subjected to an object and has a concept of that object in a black box usually referred to as the mind. Such a content of the mind is then related to the original object that is reflected in and by the mind (for short) and it is also given another form to enable the creature to communicate about his/her/its experience of that object. In case of humans, it is usually a symbol or sign, maybe that of a language which is then also related to the external object and the internal concept in the triangle of meaning (which is the same as the triangle of reference). Do not forget that as we speak of existence as inseparable from space and time, such a relationship is established in time, meaning that whoever has a concept of whatever object with whichever name will have the three inputs synchronized. And should he be not alone at that location, he/she etc. will also check that what/whom he sees as existing is real, "objective", and not "subjective" (prone to various errors) through a dialog with the members of his/her race or community.

Conceptual Content

In cognitive linguistics, abstract concepts are transformations of concrete concepts derived from embodied experience. The mechanism of transformation is structural mapping, in which properties of two or more source domains are selectively mapped onto a blended space (Fauconnier & Turner, 1995; see conceptual blending). A common class of blends are metaphors. This theory contrasts with the rationalist view that concepts are perceptions (or recollections, in Plato's term) of an independently existing world of ideas, in that it denies the existence of any such realm. It also contrasts with the

empiricist view that concepts are abstract generalizations of individual experiences, because the contingent and bodily experience is preserved in a concept, and not abstracted away. While the perspective is compatible with Jamesian pragmatism (above), the notion of the transformation of embodied concepts through structural mapping makes a distinct contribution to the problem of concept formation.

Concepts and Metaphilosophy

A long and well-established tradition philosophy posits that philosophy itself is nothing more than conceptual analysis. This view has its proponents in contemporary literature as well as historical. According to Deleuze and Guattari's What Is Philosophy? (1991), philosophy is the activity of creating concepts. This creative activity differs from previous definitions of philosophy as simple reasoning, communication or contemplation of universals. Concepts are specific to philosophy: science creates "functions", and art "sensations". A concept is always signed: thus, Descartes' Cogito or Kant's "transcendental". It is a singularity, not universal, and connects itself with others concepts, on a "plane of immanence" traced by a particular philosophy. Concepts can jump from one plane of immanence to another, combining with other concepts and therefore engaging in a "becoming-Other."

Concepts in Epistemology

Concepts are vital to the development of scientific knowledge. For example, it would be difficult to imagine physics without concepts like: energy, force, or acceleration. Concepts help to integrate apparently unrelated observations and phenomena into viable hypotheses and theories, the basic ingredients of science. The concept map is a tool that is used to help researchers visualize the inter-relationships between various concepts.

Ontology of Concepts

Although the mainstream literature in cognitive science regards the concept as a kind of mental particular, it has been suggested by some theorists that concepts are real things. In most radical form, the realist about concepts attempts to show that the supposedly mental processes are not mental at all; rather, they are abstract entities, which are just as real as any mundane object.

Plato was the starkest proponent of the realist thesis of universal concepts. By his view, concepts (and ideas in general) are innate ideas that were instantiations of a transcendental world of pure forms that lay behind the veil of the physical world. In this way, universals were explained as transcendent objects. Needless to say this form of realism was tied deeply with Plato's ontological projects. This remark on Plato is not of merely historical interest. For example, the view that numbers are Platonic objects was revived by Kurt Gödel as a result of certain puzzles that he took to arise from the phenomenological accounts.

Gottlob Frege, founder of the analytic tradition in philosophy, famously argued for the analysis of language in terms of sense and reference. For him, the sense of an expression in language describes a certain state of affairs in the world, namely, the way that some object is presented. Since many commentators view the notion of sense as identical to the notion of concept, and Frege regards senses as the linguistic representations of states of affairs in the world, it seems to follow that we may understand concepts as the manner in which we grasp the world. Accordingly, concepts (as senses) have an ontological status.

According to Carl Benjamin Boyer, in the introduction to his The History of the Calculus and its Conceptual Development, concepts in calculus do not refer to perceptions. As long as the concepts are useful and mutually compatible, they are accepted on their own. For example, the concepts of the derivative and the integral are not considered to refer to spatial or temporal perceptions of the external world of experience. Neither are they related in any way to mysterious limits in which quantities are on the verge of nascence or evanescence, that is, coming into or going out of appearance or existence. The abstract concepts are now considered to be totally autonomous, even though they originated from the process of abstracting or taking away qualities from perceptions until only the common, essential attributes remained.

Conceptual Empirical Investigations

Concepts, as abstract units of meaning, play a key role in the development and testing of theories. For example, a simple relational hypothesis can be viewed as either a conceptual hypothesis (where the abstract concepts form the meaning) or an operationalized hypothesis, which is situated in the real world by rules of interpretation. For example, take the simple hypothesis Education increases Income. The abstract notion of education and income (concepts) could have many meanings.

A conceptual hypothesis cannot be tested. They need to be converted into operational hypothesis or the abstract meaning of education must be derived or operationalized to something in the real world that can be measured. Education could be measured by “years of school completed” or “highest degree completed” etc. Income could be measured by “hourly rate of pay” or “yearly salary”, etc. The system of concepts or conceptual framework can take on many levels of complexity. When the conceptual framework is very complex and incorporates causality or explanation they are generally referred to as a theory.

The noted philosopher of science Carl Gustav Hempel says this more eloquently: “An adequate empirical interpretation turns a theoretical system into a testable theory: The hypothesis whose constituent terms have been interpreted become capable of test by reference to observable phenomena. Frequently the interpreted hypothesis will be derivative hypotheses of the theory; but their confirmation or disconfirmation by empirical data will then

immediately strengthen or weaken also the primitive hypotheses from which they were derived."

Hempel provides a useful metaphor that describes the relationship between the conceptual framework and the framework as it is observed and perhaps tested (interpreted framework): "The whole system floats, as it were, above the plane of observation and is anchored to it by rules of interpretation. These might be viewed as strings which are not part of the network but link certain points of the latter with specific places in the plane of observation. By virtue of those interpretative connections, the network can function as a scientific theory".

Memory

Conceptual Memory handles the details without being defined. It can used to store and retrieve memories in the same way we remember things. Usually with a key that is used as a search key for accessing the memory store.

Memory is implemented as a concept. It can store concepts and objects.

```
idea        memory;
memory      MyMemory;
            MyMemory = [MyMemory new];
```

To use it needs a key. To remember something.

```
[MyMemory   key: MyKey remember: MyObject] ;
```

To access something.

```
MyObject = [MyMemory key: MyKey access:];
```

To add a link between two keys use.

```
[MyMemory key: MyKey link: "favorite shows"];
```

To find something use search, it will search the key list, then any other linked lists.

```
MyObject = [MyMemory key: MyKey search:];
```

If you store two things with the same key, memory will store both and when you access the key you will get the first one on list. This would return the number of items using MyKey as a key.

```
int MyKey = [MyMemory key: MyKey keys:]
```

To forget something use.

```
[MyMemory key: MyKey forget:];
```

And it would forget last thing on list for key MyKey.

You can set up memory to work as short term or long term.

```
            [MyMemory shortTerm];       // this is the default
            [MyMemory longTerm];
```

Short Term would only remember something while the program is running. Long term will save it and it will be available the next time the program runs.

To use memory as a dictionary

```
[MyMemory key: "idea" remember: "a concept or mental impression"];
```

Then

```
MyObject = [MyMemory key: "idea" access:];
```

Would return string "a concept or mental impression". The above code is trying to work in the same way as human memory. It may change after some testing, first implementation of memory would work with text.

In psychology, memory is an organism's ability to store, retain, and recall information and experiences. Traditional studies of memory began in the fields of philosophy, including techniques of artificially enhancing memory. During the late nineteenth and early twentieth century, scientists have put memory within the paradigm of cognitive psychology. In recent decades, it has become one of the principal pillars of a branch of science called cognitive neuroscience, an interdisciplinary link between cognitive psychology and neuroscience.

Processes

From an information processing perspective there are three main stages in the formation and retrieval of memory:

Encoding or registration

Storage

Retrieval, recall or recollection

Sensory Memory

Sensory memory corresponds approximately to the initial 200–500 milliseconds after an item is perceived. The ability to look at an item, and remember what it looked like with just a second of observation, or memorisation, is an example of sensory memory. With very short presentations, participants often report that they seem to "see" more than they can actually report. The first experiments exploring this form of sensory memory were conducted by George Sperling (1960) using the "partial report paradigm". Subjects were presented with a grid of 12 letters, arranged into three rows of four. After a brief presentation, subjects were then played either a high, medium or low tone, cuing them which of the rows to report. Based on these partial report experiments, Sperling was able to show that the capacity of sensory memory was approximately 12 items, but that it degraded very quickly (within a few hundred milliseconds). Because this form of memory degrades so quickly, participants would see the display, but be unable to report all of the items (12 in the "whole report" procedure) before they decayed. This type of memory cannot be prolonged via rehearsal.

Short-Term Memory

Short-term memory allows recall for a period of several seconds to a minute without rehearsal. Its capacity is also very limited: George A. Miller (1956), when working at Bell Laboratories, conducted experiments showing that the store of short-term memory was 7±2 items (the title of his famous paper, "The magical number 7±2"). Modern estimates of the capacity of short-term memory are lower, typically on the order of 4–5 items, however, memory

capacity can be increased through a process called chunking. For example, in recalling a ten-digit telephone number, a person could chunk the digits into three groups: first, the area code (such as 215), then a three-digit chunk (123) and lastly a four-digit chunk (4567). This method of remembering telephone numbers is far more effective than attempting to remember a string of 10 digits; this is because we are able to chunk the information into meaningful groups of numbers. Herbert Simon showed that the ideal size for chunking letters and numbers, meaningful or not, was three. This may be reflected in some countries in the tendency to remember telephone numbers as several chunks of three numbers with the final four-number groups, generally broken down into two groups of two.

Short-term memory is believed to rely mostly on an acoustic code for storing information, and to a lesser extent a visual code. Conrad (1964) found that test subjects had more difficulty recalling collections of letters that were acoustically similar (e.g. E, P, D). Confusion with recalling acoustically similar letters rather than visually similar letters implies that the letters were encoded acoustically. Conrad's (1964) study however, deals with the encoding of written text, thus while memory of written language may rely on acoustic components, generalizations to all forms of memory cannot be made.

However, some individuals have been reported to be able to remember large amounts of information, quickly, and be able to recall that information in seconds.

Long-Term Memory

The storage in sensory memory and short-term memory generally have a strictly limited capacity and duration, which means that information is not retained indefinitely. By contrast, long-term memory can store much larger quantities of information for potentially unlimited duration (sometimes a whole life span). Its capacity is immeasurably large. For example, given a random seven-digit number we may remember it for only a few seconds before forgetting, suggesting it was stored in our short-term memory. On the other hand, we can remember telephone numbers for many years through repetition; this information is said to be stored in long-term memory.

While short-term memory encodes information acoustically, long-term memory encodes it semantically: Baddeley (1966) discovered that after 20 minutes, test subjects had the most difficulty recalling a collection of words that had similar meanings (e.g. big, large, great, huge).

Short-term memory is supported by transient patterns of neuronal communication, dependent on regions of the frontal lobe (especially dorsolateral prefrontal cortex) and the parietal lobe. Long-term memories, on the other hand, are maintained by more stable and permanent changes in neural connections widely spread throughout the brain. The hippocampus is essential (for learning new information) to the consolidation of information from short-term to long-term memory, although it does not seem to store information

itself. Without the hippocampus, new memories are unable to be stored into long-term memory, and there will be a very short attention span.

Furthermore, it may be involved in changing neural connections for a period of three months or more after the initial learning. One of the primary functions of sleep is thought to be improving consolidation of information, as several studies have demonstrated that memory depends on getting sufficient sleep between training and test. Additionally, data obtained from neuro imaging studies have shown activation patterns in the sleeping brain which mirror those recorded during the learning of tasks from the previous day, suggesting that new memories may be solidified through such rehearsal.

Research has suggested that long-term memory storage in humans may be regulated by DNA methylation.

Working Memory

In 1974 Baddeley and Hitch proposed a working memory model which replaced the concept of general short term memory with specific, active components. In this model, working memory consists of three basic stores: the central executive, the phonological loop and the visuo-spatial sketchpad. In 2000 this model was expanded with the multimodal episodic buffer.

The central executive essentially acts as attention. It channels information to the three component processes: the phonological loop, the visuo-spatial sketchpad, and the episodic buffer.

The phonological loop stores auditory information by silently rehearsing sounds or words in a continuous loop: the articulatory process (for example the repetition of a telephone number over and over again). A short list of data is easier to remember.

The visuospatial sketchpad stores visual and spatial information. It is engaged when performing spatial tasks (such as judging distances) or visual ones (such as counting the windows on a house or imagining images).

The episodic buffer is dedicated to linking information across domains to form integrated units of visual, spatial, and verbal information and chronological ordering (e.g., the memory of a story or a movie scene). The episodic buffer is also assumed to have links to long-term memory and semantical meaning.

The working memory model explains many practical observations, such as why it is easier to do two different tasks (one verbal and one visual) than two similar tasks (e.g., two visual), and the aforementioned word-length effect. However, the concept of a central executive as noted here has been criticised as inadequate and vague.

Levels of Processing

Craik and Lockhart (1972) proposed that it is the method and depth of processing that affects how an experience is stored in memory, rather than rehearsal.

▪ Organization - Mandler (1967) gave participants a pack of word cards and asked them to sort them into any number of piles using any system of categorisation they liked. When they were later asked to recall as many of the words as they could, those who used more categories remembered more words. This study suggested that the organization of memory is one of its central aspects (Mandler, 2011).

▪ Distinctiveness - Eysenck and Eysenck (1980) asked participants to say words in a distinctive way, e.g. spell the words out loud. Such participants recalled the words better than those who simply read them off a list.

▪ Effort - Tyler et al. (1979) had participants solve a series of anagrams, some easy (FAHTER) and some difficult (HREFAT). The participants recalled the difficult anagrams better, presumably because they put more effort into them.

▪ Elaboration - Palmere et al. (1983) gave participants descriptive paragraphs of a fictitious African nation. There were some short paragraphs and some with extra sentences elaborating the main idea. Recall was higher for the ideas in the elaborated paragraphs.

Information Type

Anderson (1976) divides long-term memory into declarative (explicit) and procedural (implicit) memories.

Declarative memory requires conscious recall, in that some conscious process must call back the information. It is sometimes called explicit memory, since it consists of information that is explicitly stored and retrieved.

Declarative memory can be further sub-divided into semantic memory, which concerns facts taken independent of context; and episodic memory, which concerns information specific to a particular context, such as a time and place. Semantic memory allows the encoding of abstract knowledge about the world, such as "Paris is the capital of France". Episodic memory, on the other hand, is used for more personal memories, such as the sensations, emotions, and personal associations of a particular place or time. Autobiographical memory - memory for particular events within one's own life - is generally viewed as either equivalent to, or a subset of, episodic memory. Visual memory is part of memory preserving some characteristics of our senses pertaining to visual experience. One is able to place in memory information that resembles objects, places, animals or people in sort of a mental image. Visual memory can result in priming and it is assumed some kind of perceptual representational system underlies this phenomenon.

In contrast, procedural memory (or implicit memory) is not based on the conscious recall of information, but on implicit learning. Procedural memory is primarily employed in learning motor skills and should be considered a subset of implicit memory. It is revealed when one does better in a given task due only to repetition - no new explicit memories have been formed, but one is unconsciously accessing aspects of those previous experiences. Procedural

memory involved in motor learning depends on the cerebellum and basal ganglia.

Topographic memory is the ability to orient oneself in space, to recognize and follow an itinerary, or to recognize familiar places. Getting lost when traveling alone is an example of the failure of topographic memory. This is often reported among elderly patients who are evaluated for dementia. The disorder could be caused by multiple impairments, including difficulties with perception, orientation, and memory.

Temporal Direction

A further major way to distinguish different memory functions is whether the content to be remembered is in the past, retrospective memory, or whether the content is to be remembered in the future, prospective memory. Thus, retrospective memory as a category includes semantic, episodic and autobiographical memory. In contrast, prospective memory is memory for future intentions, or remembering to remember (Winograd, 1988). Prospective memory can be further broken down into event- and time-based prospective remembering. Time-based prospective memories are triggered by a time-cue, such as going to the doctor (action) at 4pm (cue). Event-based prospective memories are intentions triggered by cues, such as remembering to post a letter (action) after seeing a mailbox (cue). Cues do not need to be related to the action (as the mailbox example is), and lists, sticky-notes, knotted handkerchiefs, or string around the finger are all examples of cues that are produced by people as a strategy to enhance prospective memory.

Physiology

Brain areas involved in the neuroanatomy of memory such as the hippocampus, the amygdala, the striatum, or the mammillary bodies are thought to be involved in specific types of memory. For example, the hippocampus is believed to be involved in spatial learning and declarative learning, while the amygdala is thought to be involved in emotional memory. Damage to certain areas in patients and animal models and subsequent memory deficits is a primary source of information. However, rather than implicating a specific area, it could be that damage to adjacent areas, or to a pathway traveling through the area is actually responsible for the observed deficit. Further, it is not sufficient to describe memory, and its counterpart, learning, as solely dependent on specific brain regions. Learning and memory are attributed to changes in neuronal synapses, thought to be mediated by long-term potentiation and long-term depression.

Hebb distinguished between short-term and long-term memory. He postulated that any memory that stayed in short-term storage for a long enough time would be consolidated into a long-term memory. Later research showed this to be false. Research has shown that direct injections of cortisol or epinephrine help the storage of recent experiences. This is also true for stimulation of the amygdala. This proves that excitement enhances memory by the stimulation of

hormones that affect the amygdala. Excessive or prolonged stress (with prolonged cortisol) may hurt memory storage. Patients with amygdalar damage are no more likely to remember emotionally charged words than nonemotionally charged ones. The hippocampus is important for explicit memory. The hippocampus is also important for memory consolidation. The hippocampus receives input from different parts of the cortex and sends its output out to different parts of the brain also. The input comes from secondary and tertiary sensory areas that have processed the information a lot already. Hippocampal damage may also cause memory loss and problems with memory storage.

Mind

A Brian we know about. You can open up someones head and there it is, but where is the Mind? How do people think about things? What makes us think that we understand something? A Mind is a Concept used for thinking. We combine various objects and concepts to create a Mind Concept.

Concept of Mind

The concept of mind is understood in many different ways by many different traditions, ranging from panpsychism and animism to traditional and organized religious views, as well as secular and materialist philosophies. Most agree that minds are constituted by conscious experience and intelligent thought. Common attributes of mind include perception, reason, imagination, memory, emotion, attention, and a capacity for communication. A rich set of unconscious processes are also included in many modern characterizations of mind.

Theories of mind and its function are numerous. Earliest recorded speculations are from the likes of Zoroaster, the Buddha, Plato, Aristotle, and other ancient Greek, Indian and, later, Islamic and medieval European philosophers. Pre-modern understandings of the mind, such as the neoplatonic "nous" saw it as an aspect of the soul, in the sense of being both divine and immortal, linking human thinking with the un-changing ordering principle of the cosmos itself.

Which attributes make up the mind is much debated. Some psychologists argue that only the "higher" intellectual functions constitute mind, particularly reason and memory. In this view the emotions—love, hate, fear, joy—are more primitive or subjective in nature and should be seen as different from the mind as such. Others argue that various rational and emotional states cannot be so separated, that they are of the same nature and origin, and should therefore be considered all part of what we call the mind.

In popular usage mind is frequently synonymous with thought: the private conversation with ourselves that we carry on "inside our heads." Thus we "make up our minds," "change our minds" or are "of two minds" about something. One of the key attributes of the mind in this sense is that it is a private sphere to which no one but the owner has access. No one else can "know our mind." They can only interpret what we consciously or unconsciously communicate.

Etymology

The original meaning of Old English gemynd was the faculty of memory, not of thought in general. Hence call to mind, come to mind, keep in mind, to have mind of, etc. Old English had other words to express "mind", such as hyge "mind, spirit".

The generalization of mind to include all mental faculties, thought, volition, feeling and memory, gradually develops over the 14th and 15th centuries.

The meaning of "memory" is shared with Old Norse, which has munr. The word is originally from a PIE verbal root *men-, meaning "to think, remember", whence also Latin mens "mind", Sanskrit manas "mind" and Greek μένος "mind, courage, anger".

Mental Faculties

Thought is a mental activity which allows human beings to make sense of things in the world, and to represent and interpret them in ways that are significant, or which accord with their needs, attachments, goals, commitments, plans, ends, desires, etc. Thinking involves the symbolic or semantic mediation of ideas or data, as when we form concepts, engage in problem solving, reasoning and making decisions. Words that refer to similar concepts and processes include deliberation, cognition, ideation, discourse and imagination.

Thinking is sometimes described as a "higher" cognitive function and the analysis of thinking processes is a part of cognitive psychology. It is also deeply connected with our capacity to make and use tools; to understand cause and effect; to recognize patterns of significance; to comprehend and disclose unique contexts of experience or activity; and to respond to the world in a meaningful way.

Memory is the ability to preserve, retain, and subsequently recall, knowledge, information or experience. Although memory has traditionally been a persistent theme in philosophy, the late nineteenth and early twentieth centuries also saw the study of memory emerge as a subject of inquiry within the paradigms of cognitive psychology. In recent decades, it has become one of the pillars of a new branch of science called cognitive neuroscience, a marriage between cognitive psychology and neuroscience.

The subject of memory has also been addressed in highly sophisticated ways in classic works of 20th century literature, notably in the novel of Marcel Proust, and in popular films, e.g. Eternal Sunshine of the Spotless Mind.

Imagination is the activity of generating or evoking novel situations, images, ideas or other qualia in the mind. It is a characteristically subjective activity, rather than a direct or passive experience. The term is technically used in psychology for the process of reviving in the mind percepts of objects formerly given in sense perception. Since this use of the term conflicts with that of ordinary language, some psychologists have preferred to describe this process as "imaging" or "imagery" or to speak of it as "reproductive" as opposed to "productive" or "constructive" imagination.

Things that are imagined are said to be seen in the "mind's eye". Among the many practical functions of imagination are the ability to project possible futures (or histories), to "see" things from another's perspective, and to change the way something is perceived, including to make decisions to respond to, or enact, what is imagined.

Consciousness in mammals (this includes humans) is an aspect of the mind generally thought to comprise qualities such as subjectivity, sentience, and the ability to perceive the relationship between oneself and one's environment. It is a subject of much research in philosophy of mind, psychology, neuroscience, and cognitive science. Some philosophers divide consciousness into phenomenal consciousness, which is subjective experience itself, and access consciousness, which refers to the global availability of information to processing systems in the brain. Phenomenal consciousness has many different experienced qualities, often referred to as qualia. Phenomenal consciousness is usually consciousness of something or about something, a property known as intentionality in philosophy of mind.

Mental Content

Mental contents are those items which are thought of as being "in" the mind, and capable of being formed and manipulated by mental processes and faculties. Examples include thoughts, concepts, memories, emotions, percepts and intentions. Philosophical theories of mental content include internalism, externalism, representationalism and intentionality.

Cognitive Science

In animals, the brain, or encephalon (Greek for "in the head"), is the control center of the central nervous system, responsible for thought. In most animals, the brain is located in the head, protected by the skull and close to the primary sensory apparatus of vision, hearing, equilibrioception, taste and olfaction. While all vertebrates have a brain, most invertebrates have either a centralized brain or collections of individual ganglia. Primitive animals such as sponges do not have a brain at all. Brains can be extremely complex. For example, the human brain contains more than 100 billion neurons, each linked to as many as 10,000 others.

Understanding the relationship between the brain and the mind — mind-body problem is one of the central issues in the history of philosophy — is a challenging problem both philosophically and scientifically. There are three major philosophical schools of thought concerning the answer: dualism, materialism, and idealism. Dualism holds that the mind exists independently of the brain; materialism holds that mental phenomena are identical to neuronal phenomena; and idealism holds that only mental phenomena exist.

The most straightforward scientific evidence that there is a strong relationship between the physical brain matter and the mind is the impact physical alterations to the brain have on the mind, such as with traumatic brain injury and psychoactive drug use.

In addition to the philosophical questions, the relationship between mind and brain involves a number of scientific questions, including understanding the relationship between mental activity and brain activity, the exact mechanisms by which drugs influence cognition, and the neural correlates of consciousness.

Through most of history many philosophers found it inconceivable that cognition could be implemented by a physical substance such as brain tissue (that is neurons and synapses). Philosophers such as Patricia Churchland posit that the drug-mind interaction is indicative of an intimate connection between the brain and the mind, not that the two are the same entity. Descartes, who thought extensively about mind-brain relationships, found it possible to explain reflexes and other simple behaviors in mechanistic terms, although he did not believe that complex thought, and language in particular, could be explained by reference to the physical brain alone.

Philosophy of Mind

Philosophy of mind is the branch of philosophy that studies the nature of the mind, mental events, mental functions, mental properties, consciousness and their relationship to the physical body. The mind-body problem, i.e. the relationship of the mind to the body, is commonly seen as the central issue in philosophy of mind, although there are other issues concerning the nature of the mind that do not involve its relation to the physical body.

Dualism and monism are the two major schools of thought that attempt to resolve the mind-body problem. Dualism is the position that mind and body are in some way separate from each other. It can be traced back to Plato, Aristotle and the Samkhya and Yoga schools of Hindu philosophy, but it was most precisely formulated by René Descartes in the 17th century. Substance dualists argue that the mind is an independently existing substance, whereas Property dualists maintain that the mind is a group of independent properties that emerge from and cannot be reduced to the brain, but that it is not a distinct substance.

The 20th century philosopher Martin Heidegger suggested that subjective experience and activity (i.e. the "mind") cannot be made sense of in terms of Cartesian "substances" that bear "properties" at all (whether the mind itself is thought of as a distinct, separate kind of substance or not). This is because the nature of subjective, qualitative experience is incoherent in terms of – or semantically incommensurable with the concept of – substances that bear properties. This is a fundamentally ontological argument.

Mind / Body Perspectives

Monism is the position that mind and body are not physiologically and ontologically distinct kinds of entities. This view was first advocated in Western Philosophy by Parmenides in the 5th Century BC and was later espoused by the 17th Century rationalist Baruch Spinoza. According to Spinoza's dual-aspect theory, mind and body are two aspects of an underlying reality which he variously described as "Nature" or "God".

Physicalists argue that only the entities postulated by physical theory exist, and that the mind will eventually be explained in terms of these entities as physical theory continues to evolve.

Idealists maintain that the mind is all that exists and that the external world is either mental itself, or an illusion created by the mind.

Neutral monists adhere to the position that perceived things in the world can be regarded as either physical or mental depending on whether one is interested in their relationship to other things in the world or their relationship to the perceiver. For example, a red spot on a wall is physical in its dependence on the wall and the pigment of which it is made, but it is mental in so far as its perceived redness depends on the workings of the visual system. Unlike dual-aspect theory, neutral monism does not posit a more fundamental substance of which mind and body are aspects.

The most common monisms in the 20th and 21st centuries have all been variations of physicalism; these positions include behaviorism, the type identity theory, anomalous monism and functionalism.

Many modern philosophers of mind adopt either a reductive or non-reductive physicalist position, maintaining in their different ways that the mind is not something separate from the body. These approaches have been particularly influential in the sciences, e.g. in the fields of sociobiology, computer science, evolutionary psychology and the various neurosciences. Other philosophers, however, adopt a non-physicalist position which challenges the notion that the mind is a purely physical construct.

Reductive physicalists assert that all mental states and properties will eventually be explained by scientific accounts of physiological processes and states.

Non-reductive physicalists argue that although the brain is all there is to the mind, the predicates and vocabulary used in mental descriptions and explanations are indispensable, and cannot be reduced to the language and lower-level explanations of physical science.

Continued neuroscientific progress has helped to clarify some of these issues. However, they are far from having been resolved, and modern philosophers of mind continue to ask how (if at all) the subjective qualities and the intentionality (aboutness) of mental states and properties can be explained in naturalistic terms.

Psychology

Psychology is the scientific study of human behavior, mental functioning, and experience; noology, the study of thought. As both an academic and applied discipline, Psychology involves the scientific study of mental processes such as perception, cognition, emotion, personality, as well as environmental influences, such as social and cultural influences, and interpersonal relationships, in order to devise theories of human behavior. Psychology also refers to the application of such knowledge to various spheres of human activity, including problems of individuals' daily lives and the treatment of mental health problems.

Psychology differs from the other social sciences (e.g., anthropology, economics, political science, and sociology) due to its focus on experimentation at the scale of the individual, or individuals in small groups as opposed to large groups, institutions or societies. Historically, psychology differed from biology and neuroscience in that it was primarily concerned with mind rather than brain. Modern psychological science incorporates physiological and neurological processes into its conceptions of perception, cognition, behavior, and mental disorders.

Evolutionary Psychology

Evolutionary psychology (EP) is an approach within psychology that examines psychological traits — such as memory, perception, or language — from a Darwinian evolutionary perspective. It seeks to explain how many human psychological traits are evolved adaptations, that is, the functional products of natural selection or sexual selection. Adaptationist thinking about physiological mechanisms, such as the heart, lungs, and immune system, is common in evolutionary biology. Evolutionary psychology applies the same thinking to psychology.

Evolution of the Human Mind

The evolution of human intelligence refers to a set of theories that attempt to explain how human intelligence has evolved. The question is closely tied to the evolution of the human brain, and to the emergence of human language.

The timeline of human evolution spans some 7 million years, from the separation of the Pan genus until the emergence of behavioral modernity by 50,000 years ago. Of this timeline, the first 3 million years concern Sahelanthropus, the following 2 million concern Australopithecus, while the final 2 million span the history of actual human species.

Many traits of human intelligence, such as empathy, theory of mind, mourning, ritual, and the use of symbols and tools, are already apparent in great apes although in lesser sophistication than in humans.

There is a debate between supporters of the idea of a sudden emergence of intelligence, or "Great leap forward" and those of a gradual or continuum hypothesis.

Animal intelligence

Animal cognition, or cognitive ethology, is the title given to a modern approach to the mental capacities of animals. It has developed out of comparative psychology, but has also been strongly influenced by the approach of ethology, behavioral ecology, and evolutionary psychology. Much of what used to be considered under the title of animal intelligence is now thought of under this heading. Animal language acquisition, attempting to discern or understand the degree to which animal cognition can be revealed by linguistics-related study, has been controversial among cognitive linguists.

Artificial intelligence

In 1950 Alan M. Turing published "Computing machinery and intelligence" in Mind, in which he proposed that machines could be tested for intelligence using questions and answers. This process is now named the Turing Test. The term Artificial Intelligence (AI) was first used by John McCarthy who considered it to mean "the science and engineering of making intelligent machines". It can also refer to intelligence as exhibited by an artificial (man-made, non-natural, manufactured) entity. AI is studied in overlapping fields of computer science, psychology, neuroscience and engineering, dealing with intelligent behavior, learning and adaptation and usually developed using customized machines or computers.

Research in AI is concerned with producing machines to automate tasks requiring intelligent behavior. Examples include control, planning and scheduling, the ability to answer diagnostic and consumer questions, handwriting, natural language, speech and facial recognition. As such, the study of AI has also become an engineering discipline, focused on providing solutions to real life problems, knowledge mining, software applications, strategy games like computer chess and other video games.

One of the biggest difficulties with AI is that of comprehension. Many devices have been created that can do amazing things, but critics of AI claim that no actual comprehension by the AI machine has taken place.

The debate about the nature of the mind is relevant to the development of artificial intelligence. If the mind is indeed a thing separate from or higher than the functioning of the brain, then hypothetically it would be much more difficult to recreate within a machine, if it were possible at all. If, on the other hand, the mind is no more than the aggregated functions of the brain, then it will be possible to create a machine with a recognizable mind (though possibly only with computers much different from today's), by simple virtue of the fact that such a machine already exists in the form of the human brain.

Meme

A meme is "an idea, behavior or style that spreads from person to person within a culture." A meme acts as a unit for carrying cultural ideas, symbols or practices, which can be transmitted from one mind to another through writing, speech, gestures, rituals or other imitable phenomena. Supporters of the concept regard memes as cultural analogues to genes in that they self-replicate, mutate and respond to selective pressures.

The word 'meme' is a shortening (modeled on 'gene') of 'mimeme' (from Ancient Greek μίμημα Greek pronunciation: [mí ː m ː ma] mīmēma, "something imitated", from μιμε□ σθαι mimeisthai, "to imitate", from μ□ μος mimos "mime") and it was coined by the British evolutionary biologist Richard Dawkins in The Selfish Gene (1976) as a concept for discussion of evolutionary principles in explaining the spread of ideas and cultural phenomena. Examples of memes given in the book included melodies, catch-phrases, fashion and the technology of building arches.

Advocates of the meme idea say that memes may evolve by natural selection in a manner analogous to that of biological evolution. Memes do this through the processes of variation, mutation, competition and inheritance, each of which influence a meme's reproductive success.

Memes spread through the behaviors that they generate in their hosts. Memes that propagate less prolifically may become extinct, while others may survive, spread and (for better or for worse) mutate. Memes that replicate most effectively enjoy more success. Some memes may replicate effectively even when they prove to be detrimental to the welfare of their hosts.

A field of study called memetics arose in the 1990s to explore the concepts and transmission of memes in terms of an evolutionary model. Criticism from a variety of fronts has challenged the notion that scholarship can examine memes empirically. Developments in neuroimaging may however make empirical study possible. Some commentators[who?] question the idea that one can meaningfully categorize culture in terms of discrete units.

Origins

Max Stirner's 1844 The Ego and Its Own puts forth the idea that the individual is dominated by illusory concepts ('fixed ideas' or 'spooks'), which can be shaken and undermined by each individual, though he does not use the term meme for this. He offers examples such as nationalism and religion.

Historically, the notion of a unit of social evolution, and a similar term (from Greek μνήμη mneme, “memory”), first appeared in 1904 in a work by the German Lamarckist biologist Richard Semon titled Die Mnemischen Empfindungen in ihren Beziehungen zu den Originalempfindungen (loosely translatable as “Memory-feelings in relation to original feelings”). According

to the OED, the word mneme appears in English in 1921 in L. Simon's translation of Semon's book: The Mneme.

Laurent noted the use of the term mneme in Maurice Maeterlinck's The Life of the White Ant (1926), and has highlighted similarities to Dawkins' concept.

The analogy between culturally transmitted information and genetically transmitted information was perceived clearly enough by Luigi Luca Cavalli-Sforza and Marcus W. Feldman to allow them to formulate and analyze quantitative models of cultural transmission and selection. They published a series of papers beginning in 1973.

The word meme originated with Dawkins' 1976 book The Selfish Gene. To emphasize commonality with genes, Dawkins coined the term "meme" by shortening "mimeme", which derives from the Greek word mimema ("something imitated").

Dawkins states that he did not know of the "mneme", and said that he wanted "a monosyllable that sounds a bit like 'gene'". Dawkins wrote that evolution depended not on the particular chemical basis of genetics, but only on the existence of a self-replicating unit of transmission – in the case of biological evolution, the gene.

For Dawkins, the meme exemplified another self-replicating unit with potential significance in explaining human behavior and cultural evolution.

Dawkins used the term to refer to any cultural entity that an observer might consider a replicator. He hypothesized that one could view many cultural entities as replicators, and pointed to melodies, fashions and learned skills as examples. Memes generally replicate through exposure to humans, who have evolved as efficient copiers of information and behavior. Because humans do not always copy memes perfectly, and because they may refine, combine or otherwise modify them with other memes to create new memes, they can change over time. Dawkins likened the process by which memes survive and change through the evolution of culture to the natural selection of genes in biological evolution.

Dawkins defined the meme as a unit of cultural transmission, or a unit of imitation and replication, but later definitions would vary. Memes, analogously to genes, vary in their aptitude to replicate; memes which are good at getting themselves copied tend to spread and remain, whereas the less good ones have a higher probability of being ignored and forgotten. Thus "better" memes are selected. The lack of a consistent, rigorous, and precise understanding of what typically makes up one unit of cultural transmission remains a problem in debates about memetics. In contrast, the concept of genetics gained concrete evidence with the discovery of the biological functions of DNA. Meme transmission does not necessarily require a physical medium, unlike genetic transmission.

Transmission

Life-forms can transmit information both vertically (from parent to child, via replication of genes) and horizontally (through viruses and other means). Malcolm Gladwell wrote, "A meme is an idea that behaves like a virus--that moves through a population, taking hold in each person it infects." Memes can replicate vertically or horizontally within a single biological generation. They may also lie dormant for long periods of time. Memes spread by the behaviors that they generate in their hosts. Imitation counts as an important characteristic in the propagation of memes. Imitation often involves the copying of an observed behavior of another individual, but memes may transmit from one individual to another through a copy recorded in an inanimate source, such as a book or a musical score. McNamara has suggested that memes can be thereby classified as either internal or external memes, (i-memes or e-memes). Researchers have observed memetic copying in just a few species on Earth, including hominids, dolphins and birds (that learn how to sing by imitating their parents or neighbors).

Some commentators have likened the transmission of memes to the spread of contagions. Social contagions such as fads, hysteria, copycat crime, and copycat suicide exemplify memes seen as the contagious imitation of ideas. Observers distinguish the contagious imitation of memes from instinctively contagious phenomena such as yawning and laughing, which they consider innate (rather than socially learned) behaviors.

Aaron Lynch described seven general patterns of meme transmission, or "thought contagion":

Quantity of parenthood: an idea that influences the number of children one has. Children respond particularly receptively to the ideas of their parents, and thus ideas that directly or indirectly encourage a higher birthrate will replicate themselves at a higher rate than those that discourage higher birthrates.

Efficiency of parenthood: an idea that increases the proportion of children who will adopt ideas of their parents. Cultural separatism exemplifies one practice in which one can expect a higher rate of meme-replication — because the meme for separation creates a barrier from exposure to competing ideas.

Proselytic: ideas generally passed to others beyond one's own children. Ideas that encourage the proselytism of a meme, as seen in many religious or political movements, can replicate memes horizontally through a given generation, spreading more rapidly than parent-to-child meme-transmissions do.

Preservational: ideas that influence those that hold them to continue to hold them for a long time. Ideas that encourage longevity in their hosts, or leave their hosts particularly resistant to abandoning or replacing these ideas, enhance the preservability of memes and afford protection from the competition or proselytism of other memes.

Adversative: ideas that influence those that hold them to attack or sabotage competing ideas and/or those that hold them. Adversative replication can give

an advantage in meme transmission when the meme itself encourages aggression against other memes.

Cognitive: ideas perceived as cogent by most in the population who encounter them. Cognitively transmitted memes depend heavily on a cluster of other ideas and cognitive traits already widely held in the population, and thus usually spread more passively than other forms of meme transmission. Memes spread in cognitive transmission do not count as self-replicating.

Motivational: ideas that people adopt because they perceive some self-interest in adopting them. Strictly speaking, motivationally transmitted memes do not self-propagate, but this mode of transmission often occurs in association with memes self-replicated in the efficiency parental, proselytic and preservational modes.

Memes as Discrete Units

Richard Dawkins initially defined meme as a noun that "conveys the idea of a unit of cultural transmission, or a unit of imitation". John S. Wilkins retained the notion of meme as a kernel of cultural imitation while emphasizing the meme's evolutionary aspect, defining the meme as "the least unit of sociocultural information relative to a selection process that has favourable or unfavourable selection bias that exceeds its endogenous tendency to change."

The meme as a unit provides a convenient means of discussing "a piece of thought copied from person to person", regardless if that thought contains others inside it, or forms part of a larger meme. A meme could consist of a single word, or a meme could consist of the entire speech in which that word first occurred. This forms an analogy to the idea of a gene as a single unit of self-replicating information found on the self-replicating chromosome.

While the identification of memes as "units" conveys their nature to replicate as discrete, indivisible entities, it does not imply that thoughts somehow become quantized or that "atomic" ideas exist that cannot be dissected into smaller pieces. A meme has no given size. Susan Blackmore writes that melodies from Beethoven's symphonies are commonly used to illustrate the difficulty involved in delimiting memes as discrete units. She notes that while the first four notes of Beethoven's Fifth Symphony (listen (help·info)) form a meme widely replicated as an independent unit, one can regard the entire symphony as a single meme as well.

The inability to pin an idea or cultural feature to quantifiable key units is widely acknowledge as a problem for memetics. It has been argued however that the traces of memetic processing can be quantified utilizing neuroimaging techniques which measure changes in the connectivity profiles between brain regions." Blackmore meets such criticism by stating that memes compare with genes in this respect: that while a gene has no particular size, nor can we ascribe every phenotypic feature directly to a particular gene, it has value because it encapsulates that key unit of inherited expression subject to evolutionary pressures. To illustrate, she notes evolution selects for the gene

for features such as eye color; it does not select for the individual nucleotide in a strand of DNA. Memes play a comparable role in understanding the evolution of imitated behaviors.

The 1981 book Genes, Mind, and Culture: The Coevolutionary Process by Charles J. Lumsden and E. O. Wilson proposed the theory that genes and culture co-evolve, and that the fundamental biological units of culture must correspond to neuronal networks that function as nodes of semantic memory. They coined their own term, "culturgen", which did not catch on. Coauthor Wilson later acknowledged the term meme as the best label for the fundamental unit of cultural inheritance in his 1998 book Consilience: The Unity of Knowledge, which elaborates upon the fundamental role of memes in unifying the natural and social sciences.

Evolutions Influences on Memes

Richard Dawkins noted the three conditions that must exist for evolution to occur:

variation, or the introduction of new change to existing elements; heredity or replication, or the capacity to create copies of elements; differential "fitness", or the opportunity for one element to be more or less suited to the environment than another.

Dawkins emphasizes that the process of evolution naturally occurs whenever these conditions co-exist, and that evolution does not apply only to organic elements such as genes. He regards memes as also having the properties necessary for evolution, and thus sees meme evolution as not simply analogous to genetic evolution, but as a real phenomenon subject to the laws of natural selection. Dawkins noted that as various ideas pass from one generation to the next, they may either enhance or detract from the survival of the people who obtain those ideas, or influence the survival of the ideas themselves. For example, a certain culture may develop unique designs and methods of tool-making that give it a competitive advantage over another culture. Each tool-design thus acts somewhat similarly to a biological gene in that some populations have it and others do not, and the meme's function directly affects the presence of the design in future generations. In keeping with the thesis that in evolution one can regard organisms simply as suitable "hosts" for reproducing genes, Dawkins argues that one can view people as "hosts" for replicating memes.

Consequently, a successful meme may or may not need to provide any benefit to its host. Unlike genetic evolution, memetic evolution can show both Darwinian and Lamarckian traits. Cultural memes will have the characteristic of Lamarckian inheritance when a host aspires to replicate the given meme through inference rather than by exactly copying it. Take for example the case of the transmission of a simple skill such as hammering a nail, a skill that a learner imitates from watching a demonstration without necessarily imitating every discrete movement modeled by the teacher in the demonstration, stroke

for stroke. Susan Blackmore distinguishes the difference between the two modes of inheritance in the evolution of memes, characterizing the Darwinian mode as "copying the instructions" and the Lamarckian as "copying the product."

Cell

A Cell is just like a Virtual Computer and can run programs written in C Code or Machine Token Codes.

The cell is the functional basic unit of life. It was discovered by Robert Hooke and is the functional unit of all known living organisms. It is the smallest unit of life that is classified as a living thing, and is often called the building block of life. Organisms can be classified as unicellular (consisting of a single cell; including most bacteria) or multicellular (including plants and animals). Humans contain about 100 trillion cells; a typical cell size is 10 µm and a typical cell mass is 1 nanogram. The longest human cells are about 135 µm in the anterior horn in the spinal cord while granule cells in the cerebellum, the smallest, can be some 4 µm and the longest cell can reach from the toe to the lower brain stem (Pseudounipolar cells). The largest known cells are unfertilised ostrich egg cells, which weigh 3.3 pounds.

In 1835, before the final cell theory was developed, Jan Evangelista Purkyně observed small "granules" while looking at the plant tissue through a microscope. The cell theory, first developed in 1839 by Matthias Jakob Schleiden and Theodor Schwann, states that all organisms are composed of one or more cells, that all cells come from preexisting cells, that vital functions of an organism occur within cells, and that all cells contain the hereditary information necessary for regulating cell functions and for transmitting information to the next generation of cells.

The word cell comes from the Latin cellula, meaning "a small room". The descriptive term for the smallest living biological structure was coined by Robert Hooke in a book he published in 1665 when he compared the cork cells he saw through his microscope to the small rooms monks lived in.

Anatomy

There are two types of cells: eukaryotic and prokaryotic. Prokaryotic cells are usually independent, while eukaryotic cells are often found in multicellular organisms.

Prokaryotic Cells

The prokaryote cell is simpler, and therefore smaller, than a eukaryote cell, lacking a nucleus and most of the other organelles of eukaryotes. There are two kinds of prokaryotes: bacteria and archaea; these share a similar structure.

Nuclear material of prokaryotic cell consist of a single chromosome that is in direct contact with cytoplasm. Here, the undefined nuclear region in the cytoplasm is called nucleoid.

Eukaryotic Cells

Plants, animals, fungi, slime moulds, protozoa, & algae are all Eukaryotic. These cells are about 15 times wider than a typical prokaryote and can be as much as 1000 times greater in volume. The major difference between prokaryotes and eukaryotes is that eukaryotic cells contain membrane-bound compartments in which specific metabolic activities take place. Most important among these is a cell nucleus, a membrane-delineated compartment that houses the eukaryotic cell's DNA. This nucleus gives the eukaryote its name, which means "true nucleus." Other differences include:

The plasma membrane resembles that of prokaryotes in function, with minor differences in the setup. Cell walls may or may not be present.

The eukaryotic DNA is organized in one or more linear molecules, called chromosomes, which are associated with histone proteins. All chromosomal DNA is stored in the cell nucleus, separated from the cytoplasm by a membrane. Some eukaryotic organelles such as mitochondria also contain some DNA. Eukaryotes can move using motile cilia or flagella. The flagella are more complex than those of prokaryotes.

Genetic Material

Two different kinds of genetic material exist: deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). Most organisms use DNA for their long-term information storage, but some viruses (e.g., retroviruses) have RNA as their genetic material. The biological information contained in an organism is encoded in its DNA or RNA sequence. RNA is also used for information transport (e.g., mRNA) and enzymatic functions (e.g., ribosomal RNA) in organisms that use DNA for the genetic code itself. Transfer RNA (tRNA) molecules are used to add amino acids during protein translation.

A human cell has genetic material contained in the cell nucleus (the nuclear genome) and in the mitochondria (the mitochondrial genome). In humans the nuclear genome is divided into 23 pairs of linear DNA molecules called chromosomes. The mitochondrial genome is a circular DNA molecule distinct from the nuclear DNA.

Origin of the First Cell

The origin of cells has to do with the origin of life, which began the history of life on Earth. There are several theories about the origin of small molecules that could lead to life in an early Earth. One is that they came from meteorites. Another is that they were created at deep-sea vents. A third is that they were synthesized by lightning in a reducing atmosphere; although it is not clear if Earth had such an atmosphere.

There are essentially no experimental data defining what the first self-replicating forms were. RNA is generally assumed the earliest self-replicating molecule, as it is capable of both storing genetic information and catalyzing chemical reactions (see RNA world hypothesis). But some other entity with the potential to self-replicate could have preceded RNA, like clay or peptide

nucleic acid. Cells emerged at least 4.0 billion years ago. The current belief is that these cells were heterotrophs.

Computer Cell

Is made up of a Mind Concept and a Object Body, it also has the ability to copy itself. The Mind can run a program in the same way a “VIrtual Machine” would. Part of the Body definition would be the “Virtual Machine” that the Mind Concept will use. The Body could also have the ability to move to a new network location or processor.

Soul

“In attempting to construct such machines we should not be irreverently usurping His power of creating souls.” Alan Turing

The Conceptual Soul of the machine is used for an internal dialogue, so as our Mind concept can talk to itself and practice conversation. This allows for improvements in dialogue to occur.

The Soul can also control goals and emotions and provide likes and dis-likes used in judgements and hunches. The Soul Concept can be used when a Mind concept wants to practice scenarios.

A soul – in certain spiritual, philosophical, and psychological traditions – is the incorporeal essence of a person or living thing or object. There is absolutely no scientific proof of souls existing - yet many philosophical and spiritual systems still teach that humans have souls, and others teach that all living things and even inanimate objects (such as rivers) have souls. The latter belief is commonly called animism. Soul sometimes functions as a synonym for spirit, mind or self.

Etymology

The Modern English word soul derived from Old English sáwol, sáwel, first attested to in the 8th century poem Beowulf v. 2820 and in the Vespasian Psalter 77.50, and is cognate with other Germanic and Baltic terms for the same idea, including Gothic saiwala, Old High German sêula, sêla, Old Saxon sêola, Old Low Franconian sêla, sîla, Old Norse sála as well as Lithuanian siela. Further etymology of the Germanic word is uncertain. A more recent suggestion connects it with a root for "binding", Germanic *sailian (OE sēlian, OHG seilen), related to the notion of being "bound" in death, and the practice of ritually binding or restraining the corpse of the deceased in the grave to prevent his or her return as a ghost.

The word is probably an adaptation by early missionaries—particularly Ulfilas, apostle to the Goths during the 4th century—of a native Germanic concept, which was a translation of Greek ψυχή psychē "life, spirit, consciousness".

The Greek word is derived from a verb "to cool, to blow" and hence refers to the vital breath, the animating principle in humans and other animals, as opposed to soma meaning "body". It could refer to a ghost or spirit of the dead in Homer, and to a more philosophical notion of an immortal and immaterial essence left over at death since Pindar. Latin anima figured as a translation of since Terence. Psychē occurs juxtaposed to e.g. in Matthew 10:28:

Authorized King James Version (KJV) "And fear not them which kill the body, but are not able to kill the soul: but rather fear Him which is able to destroy both soul and body in hell."

In the Septuagint (LXX) translates Hebrew nephesh, meaning "life, vital breath", which is in English variously translated as "soul, self, life, creature, person, appetite, mind, living being, desire, emotion, passion"; e.g. in Genesis 1:20:

And God created great whales, and every living creature that moveth."

Paul of Tarsus distinguishs between the Jewish notions of nephesh and ruah (spirit) (also in LXX, e.g. Genesis 1:2 spiritus Dei = "the Spirit of God").

Semantics

Although the terms soul and spirit are sometimes used interchangeably, soul may denote a more worldly and less transcendent aspect of a person. According to psychologist James Hillman, soul has an affinity for negative thoughts and images, whereas spirit seeks to rise above the entanglements of life and death. The words soul and psyche can also be treated synonymously, although psyche has more physical connotations, whereas soul is connected more closely to spirituality and religion.

Philosophical views

The Ancient Greeks used the same word for 'alive' as for 'ensouled', indicating that the earliest surviving western philosophical view found the terms soul and aliveness were synonymous - perhaps not that having life universally presupposed the possession of a soul as in Buddhism, but that full "aliveness" and the soul were conceptually linked.

Socrates and Plato

Plato, drawing on the words of his teacher Socrates, considered the soul the essence of a person, being that which decides how we behave. He considered this essence to be an incorporeal, eternal occupant of our being. As bodies die the soul is continually reborn in subsequent bodies. The Platonic soul comprises three parts:

the logos (mind, nous, or reason)

the thymos (emotion, or spiritedness, or masculine)

the eros (appetitive, or desire, or feminine)

Each of these has a function in a balanced, level and peaceful soul.

Aristotle

Aristotle defined the soul or psyche as the essence or definition of a living being, but argued against its having a separate existence from the physical body. In Aristotle's view, the primary activity of a living thing constitutes its soul; for example, the soul of an eye, if it were an independent organism, would be seeing (its purpose or final cause). By an imperfect analogy, an

artifact, such as a knife or axe, (which has a clear purpose), if it had a soul, that soul would be the act of cutting, because 'cutting' is, in essence, what it is to be a knife. Unlike Plato and the medieval religious tradition, Aristotle did not consider the soul to be a separate, immortal occupant of the body; just as the act of cutting cannot occur without a blade, the soul ceases to exist at the death of the body. In his view, the soul is the actuality of a living body. More precisely, the soul is the "first actuality" of a body, in so far as it has the capacity to be alive.

Aristotle used his theory of the soul in many of his works; most notably De Anima (On the Soul). Aristotle divided the intellectual faculty into two principal parts, the "deliberative" or "calculative" and the "scientific" or "theoretical". As these are parts of the rational faculty of man, their correct activity also constitutes the "excellences" or "virtues" of the rational part of man, of which there are five: art; prudence and science, corresponding in name to the faculties themselves; nous, often translated as "understanding" or "intelligence"; and sophia, or "wisdom". Nous is rational and intuitive comprehension of either a priori or axiomatic first principles and sophia combines this understanding with science.

Avicenna and Ibn Al-Nafis

Following Aristotle, the Persian Muslim philosopher-physician, Avicenna (Ibn Sina) and Arab philosopher Ibn al-Nafis, further elaborated on the Aristotelian understanding of the soul and developed their own theories on the soul. They both made a distinction between the soul and the spirit, and in particular, the Avicennian doctrine on the nature of the soul was influential among the Scholastics. Some of Avicenna's views on the soul included the idea that the immortality of the soul is a consequence of its nature, and not a purpose for it to fulfill. In his theory of "The Ten Intellects", he viewed the human soul as the tenth and final intellect.

While he was imprisoned, Avicenna wrote his famous "Floating Man" thought experiment to demonstrate human self-awareness and the substantiality of the soul. He told his readers to imagine themselves suspended in the air, isolated from all sensations, which includes no sensory contact with even their own bodies. He argues that, in this scenario, one would still have self-consciousness. He thus concludes that the idea of the self is not logically dependent on any physical thing, and that the soul should not be seen in relative terms, but as a primary given, a substance. This argument was later refined and simplified by René Descartes in epistemic terms when he stated: "I can abstract from the supposition of all external things, but not from the supposition of my own consciousness."

Avicenna generally supported Aristotle's idea of the soul originating from the heart, whereas Ibn al-Nafis rejected this idea and instead argued that the soul "is related to the entirety and not to one or a few organs". He further criticized Aristotle's idea that every unique soul requires the existence of a unique

source, in this case the heart. Ibn al-Nafis concluded that "the soul is related primarily neither to the spirit nor to any organ, but rather to the entire matter whose temperament is prepared to receive that soul" and he defined the soul as nothing other than "what a human indicates by saying 'I'".

Thomas Aquinas

Following Aristotle and Avicenna, St. Thomas Aquinas understood the soul to be the first principle, or act, of the body. However, his epistemological theory required that, since the intellectual soul is capable of knowing all material things, and since in order to know a material thing there must be no material thing within it, the soul was definitely not corporeal. Therefore, the soul had an operation separate from the body and therefore could subsist without the body. Furthermore, since the rational soul of human beings was subsistent and was not made up of matter and form, it could not be destroyed in any natural process. The full argument for the immortality of the soul and Thomas's elaboration of Aristotelian theory is found in Question 75 of the Summa Theologica.

Immanuel Kant

In his discussions of rational psychology Immanuel Kant (1724–1804) identified the soul as the "I" in the strictest sense and that the existence of inner experience can neither be proved nor disproved. "We cannot prove a priori the immateriality of the soul, but rather only so much: that all properties and actions of the soul cannot be cognized from materiality." It is from the "I", or soul, that Kant proposes transcendental rationalization, but cautions that such rationalization can only determine the limits of knowledge if it is to remain practical.

James Hillman

Contemporary psychology is defined as the study of mental processes and behavior. However, the word "psychology" literally means "study of the soul", and psychologist James Hillman, the founder of archetypal psychology, has been credited with "restoring 'soul' to its psychological sense."

Although the words soul and spirit are often viewed as synonyms, Hillman argues that they can refer to antagonistic components of a person. Hillman believes that religion—especially monotheism and monastic faiths—and humanistic psychology have tended to the spirit, often at the unfortunate expense of soul.

Archetypal psychology acknowledges this third position by attuning to, and often accepting, the archetypes, dreams, myths, and even psychopathologies through which soul, in Hillman's view, expresses itself.

Philosophy of Mind

For a contemporary understanding of the soul/mind and the problem concerning its connection to the brain/body, consider the rejection of Descartes'

mind/body dualism by Gilbert Ryle's ghost-in-the-machine argument, [clarification needed] the tenuous unassailability of Richard Swinburne's argument for the soul,[clarification needed] and the advances, which have been made in neuroscience and which are steadily uncovering the truth/falsity[vague] of the concept of an independent soul/mind. The philosophies mind and of personal identity also contribute to a contemporary understanding of the mind. The contemporary approach does not so much attack the existence of an independent soul as render the concept less relevant. The advances in neuroscience mainly serve to support the mind/brain identity hypothesis, showing the extent of the correlation between mental states and physical-brain states. The notion of soul has less explanatory power in a western world-view which prefers the empirical explanations involving observable and locatable elements of the brain. Even so, there remain considerable objections to simple-identity theory. Notably, philosophers such as Thomas Nagel and David Chalmers have argued that the correlation between physical-brain states and mental states is not strong enough to support identity theory. Nagel (1974) argues that no amount of physical data is sufficient to provide the "what it is like" of first-person experience, and Chalmers (1996) argues for an "explanatory gap" between functions of the brain and phenomenal experience. On the whole, brain/mind identity theory does poorly in accounting for mental phenomena of qualia and intentionality. While neuroscience has done much to illuminate the functioning of the brain, much of subjective experience remains mysterious.

Buddhism

Buddhism teaches that all things are in a constant state of flux: all is changing, and no permanent state exists by itself. This applies to human beings as much as to anything else in the cosmos. Thus, a human being has no permanent self. According to this doctrine of anatta (Pāli; Sanskrit: anātman) — "no-self" or "no soul" — the words "I" or "me" do not refer to any fixed thing. They are simply convenient terms that allow us to refer to an ever-changing entity.

The anatta doctrine is not a kind of materialism. Buddhism does not deny the existence of "immaterial" entities, and it (at least traditionally) distinguishes bodily states from mental states. Thus, the conventional translation of anatta as "no-soul" can be confusing. If the word "soul" simply refers to an incorporeal component in living things that can continue after death, then Buddhism does not deny the existence of the soul. Instead, Buddhism denies the existence of a permanent entity that remains constant behind the changing corporeal and incorporeal components of a living being. Just as the body changes from moment to moment, so thoughts come and go. And there is no permanent, underlying mind that experiences these thoughts, as in Cartesianism; rather, conscious mental states simply arise and perish with no "thinker" behind them. When the body dies, the incorporeal mental processes continue and are reborn in a new body. Because the mental processes are constantly changing, the

being that is reborn is neither entirely different than, nor exactly the same as, the being that died.

However, the new being is continuous with the being that died — in the same way that the "you" of this moment is continuous with the "you" of a moment before, despite the fact that you are constantly changing.

Buddhist teaching holds that a notion of a permanent, abiding self is a delusion that is one of the causes of human conflict on the emotional, social, and political levels.

They add that an understanding of anatta provides an accurate description of the human condition, and that this understanding allows us to pacify our mundane desires.

Various schools of Buddhism have differing ideas about what continues after death. The Yogacara school in Mahayana Buddhism said there are Store consciousness which continue to exist after death. In some schools, particularly Tibetan Buddhism, the view is that there are three minds: very subtle mind, which does not disintegrate in death; subtle mind, which disintegrates in death and which is "dreaming mind" or "unconscious mind"; and gross mind, which does not exist when one is sleeping. Therefore, gross mind less permanent than subtle mind, which does not exist in death. Very subtle mind, however, does continue, and when it "catches on", or coincides with phenomena, again, a new subtle mind emerges, with its own personality/assumptions/habits, and that entity experiences karma in the current continuum.

Certain modern Buddhists, particularly in Western countries, reject—or at least take an agnostic stance toward—the concept of rebirth or reincarnation, which they view as incompatible with the concept of anatta. Stephen Batchelor discusses this issue in his book, Buddhism Without Beliefs. Others point to research that has been conducted at the University of Virginia as proof that some people are reborn.

Judaism

The Hebrew terms nephesh, ruach (literally "wind"), and neshama (literally "breath") are used to describe the soul or spirit. The soul is believed to be given by God to a person by his/her first breath, as mentioned in Genesis, "And the LORD God formed man [of] the dust of the ground, and breathed into his nostrils the breath of life; and man became a living soul." (Genesis 2:7). From this statement, the rabbinical interpretation is often that human embryos do not have souls, though the orthodox often oppose abortion as a form of birth control. Judaism relates the quality of one's soul to one's performance of mitzvot and reaching higher levels of understanding, and thus closeness to God. A person with such closeness is called a tzadik. Judaism also has a concept of purity of body and soul, which requires avoidance of "unclean" things. Such practices mentioned in the Torah include the keeping of kashrut and daily bathing (tevilah) in a mikveh. In biblical times, it was believed that

"impurity" was something that could be spread by touching, and unclean people were temporarily separated from the group. Though Jewish theology does not agree on the nature of an afterlife, the soul is said to "return to God" after death.

Kabbalah and other mystic traditions go into greater detail into the nature of the soul. Kabbalah separates the soul into three elements: the nephesh is related to instinct, the ruach is related to morality, and the neshamah is related to intellect and the awareness of God. Kabbalah furthermore proposed a concept of reincarnation, the gilgul.

Christianity

Soul carried to Heaven by William Bouguereau

The Christian view of the soul is based upon the teaching of both the Old Testament and New Testament. The Old Testament contains the statements "Then shall the dust return to the earth as it was: and the spirit shall return unto God who gave it" (Ecclesiastes 12:7) and "And the LORD God formed man [of] the dust of the ground, and breathed into his nostrils the breath of life; and man became a living soul." (Genesis 2:7). In the New Testament can be found a statement by Paul the Apostle, "And so it is written, the first man Adam was made a living soul; the last Adam [was made] a quickening spirit." (1 Corinthians 15:45).

Most Christians understand the soul as an ontological reality distinct from, yet integrally connected with, the body. Its characteristics are described in moral, spiritual, and philosophical terms. When people die, their souls will be judged by God and determined to spend an eternity in heaven or in hell. Though all branches of Christianity –Catholics, Eastern Orthodox and Oriental Orthodox, Evangelical or mainline Protestants – teach that Jesus Christ plays a decisive role in the salvation process, the specifics of that role and the part played by individual persons or ecclesiastical rituals and relationships, is a matter of wide diversity in official church teaching, theological speculation and popular practice. Some Christians believe that if one has not repented of one's sins and trusted in Jesus Christ as Lord and Saviour, one will go to hell and suffer eternal separation from God. Variations also exist on this theme, e.g. some which hold that the unrighteous soul will be destroyed instead of suffering eternally (Annihilationism). Believers will inherit eternal life in heaven and enjoy eternal fellowship with God. There is also a belief that babies (including the unborn) and those with cognitive or mental impairments who have died will be received into heaven on the basis of God's grace through the sacrifice of Jesus.

Roman Catholic Beliefs

Among Christians, there is uncertainty regarding whether human embryos have souls, and at what point between conception and birth the fetus acquires a soul and consciousness. This uncertainty is the general reasoning behind many Christians' belief that abortion should not be legal.

The present Catechism of the Catholic Church defines the soul as "the innermost aspect of humans, that which is of greatest value in them, that by which they are most especially in God's image: 'soul' signifies the spiritual principle in man." All souls living and dead will be Judged by Jesus Christ when he comes back to earth.

The souls of those who die unrepentant of serious sins, or in conscious rejection of God, will at judgment day may be forever in a state called Hell. The Catholic Church teaches that the existence of each individual soul is dependent wholly upon God: "The doctrine of the faith affirms that the spiritual and immortal soul is created immediately by God."

Hinduism

In Hinduism, the Sanskrit words most closely corresponding to soul are "Jeeva", "Atman" and "Purusha", meaning the individual Self. The term "soul" is misleading as it implies an object possessed, whereas Self signifies the subject which perceives all objects. This self is held to be distinct from the various mental faculties such as desires, thinking, understanding, reasoning and self-image (ego), all of which are considered to be part of Prakriti (nature).

In Bhagavad - Gita 2.20 Lord Krishna describes the soul in the following way:

"For the soul there is neither birth nor death at any time. He has not come into being, does not come into being, and will not come into being. He is unborn, eternal, ever - existing and primeval. He is not slain when the body is slain."

Srila Prabhupada, a great Vaishnava saint of the modern time further explains: The soul does not take birth there, and the soul does not die...And because the soul has no birth, he therefore has no past, present or future. He is eternal, ever-existing and primeval - that is, there is no trace in history of his coming into being.

Since the quality of Atman is primarily consciousness, all sentient and insentient beings are pervaded by Atman, including plants, animals, humans and gods. The difference between them is the contracted or expanded state of that consciousness. For example, animals and humans share in common the desire to live, fear of death, desire to procreate and to protect their families and territory and the need for sleep, but animals' consciousness is more contracted and has less possibility to expand than does human consciousness.

When the Atman becomes embodied it is called birth, when the Atman leaves a body it is called death. The Atman transmigrates from one body to another body based on karmic [performed deeds] reactions.

Islam

There is a hadith reported by Abd Allah ibn Mas'ud, in which it is stated that the soul is put into the human embryo 40 days after fertilization takes place. This version of hadith is supported by some other hadiths narrated by Sahih al-Bukhari and Sahih Al Muslim.

The hadith is as follows: "Verily the creation of every one of you is brought together in the mother's womb as a drop of semen for forty days, then it becomes a clot for the same period, then it becomes a blob of flesh for the same period. Then the angel will be sent unto it to blow into it a SPIRIT, and the angel is ordered (to carry out) with four instructions, to write down its livelihood, the span of life, its deeds, and whether it is wretched or fortunate."

According to the Quran, Ruh (Soul) is a command from Allah (God).

"They put questions to you concerning the Spirit. Say the Spirit is at my Lord's command. ..and of knowledge only a meagre part has been imparted to you." (Qur'an 17:85)

Taoism

According to Chinese traditions, every person has two types of soul called hun and po, which are respectively yang and yin. Taoism believes in ten souls, sanhunqipo "three hun and seven po". The pò is linked to the dead body and the grave, whereas the hún is linked to the ancestral tablet. A living being that loses any of them is said to have mental illness or unconsciousness, while a dead soul may reincarnate to a disability, lower desire realms or may even be unable to reincarnate. Also, Journeys to the Under-World said there can be hundreds of divisible souls.

Zoroastrianism

In theological reference to the soul, the terms "life" and "death" are viewed as emphatically more definitive than the common concepts of "biological life" and "biological death". Because the soul is said to be transcendent of the material existence, and is said to have (potentially) eternal life, the death of the soul is likewise said to be an eternal death. Thus, in the concept of divine judgment, God is commonly said to have options with regard to the dispensation of souls, ranging from Heaven (i.e. angels) to hell (i.e. demons), with various concepts in between. Typically both Heaven and hell are said to be eternal, or at least far beyond a typical human concept of lifespan and time.

Some transhumanists believe that it will become possible to perform mind transfer, either from one human body to another, or from a human body to a computer. Operations of this type (along with teleportation), raise philosophical questions related to the concept of the soul.

Spirituality and New Age

In Helena Blavatsky's Theosophy the soul is the field of our psychological activity (thinking, emotions, memory, desires, will, and so on) as well as of the so-called paranormal or psychic phenomena (extrasensory perception, out-of-body experiences, etc.). However, the soul is not the highest, but a middle dimension of human beings. Higher than the soul is the spirit, which is considered to be the real self; the source of everything we call “good”—

happiness, wisdom, love, compassion, harmony, peace, etc. While the spirit is eternal and incorruptible, the soul is not. The soul acts as a link between the material body and the spiritual self, and therefore shares some characteristics of both. The soul can be attracted either towards the spiritual or towards the material realm, being thus the “battlefield” of good and evil. It is only when the soul is attracted towards the spiritual and merges with the Self that it becomes eternal and divine.

Rudolf Steiner differentiated three stages of soul development, which interpenetrate one another in consciousness:

an initial centering on sensations, drives, and passions, with strong conative (will) and emotional components, which he termed the "sentient soul"; a developing focus on internalizations of and reflections on outer experience, with strong affective (feeling) and cognitive (thinking) components, which he termed the "intellectual" or "mind soul"; and a search for factors not dependent upon personal considerations, objective truths, which he termed the "consciousness soul".

Some people, who do not necessarily favor organized religions, simply label themselves as "spiritual" and hold that both humans and all other living creatures have souls. Some further believe the entire universe has a cosmic soul as a spirit or unified consciousness. Such a conception of the soul may link with the idea of an existence before and after the present one, and one could consider such a soul as the spark, or the self, the "I" in existence that feels and lives life.

In Surat Shabda Yoga, the soul is considered to be an exact replica and spark of the Divine. The purpose of Surat Shabd Yoga is to realize one's True Self as soul (Self-Realisation), True Essence (Spirit-Realisation) and True Divinity (God-Realisation) while living in the physical body.

George Gurdjieff in his Fourth Way taught that nobody is ever born with a soul. Rather, an individual must create a soul[vague] during the course of their life. Without a soul, Gurdjieff taught that one will "die like a dog".

Eckankar, founded by Paul Twitchell in 1965, defines Soul as the true self; the inner, most sacred part of each person.

The Indian spiritual teacher Meher Baba also held that the nature of the soul is divine:

God is indivisibly in everyone and everything. The apparent separateness is due to ignorance. A drop in the ocean is one with the ocean, and as soon as a bubble forms over it, it becomes separate from the ocean. And when the bubble bursts, the drop is again one with the ocean. In the same way the soul, when it is covered by the bubble of mind, energy and matter, becomes separate from God. But as soon as the bubble is burst, which is when the ego-self is annihilated through love for God, it finds that it is One with God.

Science

Science and medicine seek naturalistic accounts of the observable natural world. This stance is known as methodological naturalism.

Much of the scientific study relating to the soul has involved investigating the soul as an object of human belief, or as a concept that shapes cognition and an understanding of the world, rather than as an entity in and of itself.

When modern scientists speak of the soul outside of this cultural and psychological context, they generally treat soul as a poetic synonym for mind. Francis Crick's book, The Astonishing Hypothesis, for example, has the subtitle, "The scientific search for the soul". Crick held the position that one can learn everything knowable about the human soul by studying the workings of the human brain. Depending on one's belief regarding the relationship between the soul and the mind, then, the findings of neuroscience may be relevant to one's understanding of the soul. Skeptic Robert T. Carroll suggests that the concept of a non-substantial substance is an oxymoron, and that the scholarship done by philosophers and psychologists based on the assumption of a non-physical entity has not furthered scientific understanding of the working of the mind.

To extend the above hypothesis, one can argue that every kind of matter or energy exchange between a human biological system and its environment can be attributed to a series of complex physical and chemical transformations that occur inside the body and are largely controlled by the brain. Hence many phenotypic, genotypic, behavioral and emotional characteristics or states of a human can be identified as bearing physio-chemical cause.

For example, solar radiation carries electromagnetic energy that causes the accumulation of the pigment melanin in skin cells, which is perceived as tanning.

Neurochemicals (most famously endorphins) are responsible for what can be described as feelings of well being, love and pain. Similarly, memory can be seen as an atomic reconstruction of an image in the brain at the expense of chemical energy taken in by food.

More social or long-term characteristics such as one's beliefs may be regarded as a combination of chemical concentrations and molecular arrangements, organized in a kind of chemical recipe formulated over the course of years or millennia. An early precursor of this recipe is the primordial soup. Crudely put, the soul can be defined as chemical information exhibited on living matter.

A strict line of causality fails to explain certain phenomenon within human experience such as free will, which have at times been attributed to the soul. (See also: Determinism and free will)

In his book Consilience, E. O. Wilson took note that sociology has identified belief in a soul as one of the universal human cultural elements. Wilson suggested that biologists need to investigate how human genes predispose people to believe in a soul.

Daniel Dennett has championed the idea that the human survival strategy depends heavily on adoption of the intentional stance, a behavioral strategy that predicts the actions of others based on the expectation that they have a mind like one's own (see theory of mind). Mirror neurons in brain regions such as Broca's area may facilitate this behavioral strategy. The intentional stance, Dennett suggests, has proven so successful that people tend to apply it to all aspects of human experience, thus leading to animism and to other conceptualizations of soul.

The word-concept Soul has a secular and non-secular aspect. To integrate the two in a coherent statement Soul would be defined as: The interaction of mind, body and spirit reflecting through conscience the appropriateness of individual or collective behavior. Through a connection to the Soul the mind apprehends abstractions implicit in spirit whether that be of transcendent derivation or temporal analysis.

Understanding

Understanding (also called intellection) is a psychological process related to an abstract or physical object, such as a person, situation, or message whereby one is able to think about it and use concepts to deal adequately with that object. Understanding is a relation between the knower and an object of understanding. Understanding implies abilities and dispositions with respect to an object of knowledge sufficient to support intelligent behavior. An understanding is the limit of a conceptualization. To understand something is to have conceptualized it to a given measure.

Is Understanding Definable?

Yes, albeit with difficulty. The easiest way to define understanding is to do so in respect of specific relationships. For examples one can define the concept in the context of trust between two individuals.

A good example is the definition proposed by a Doctoral researcher from the University of Cranfield UK. In his analysis of "understanding as an antecedents of trust in virtual organisations, Joel De Messan (2011) proposed the following explanation of Understanding. "To understand someone or something, is to possess enough information about the person or thing to be able to accurately explain their unique behaviours and characteristics, accommodate their differences, or display tolerance and compassion in one's actions or judgement towards them on the basis of the insight that the information possessed provides into their reality". He therefore defined understanding as "The successful sense-making or accurate synthesis of information relating to an entity (a person, an object, a concept, or a phenomenon) that permits a justified explanation of the characteristics, behaviors and events associated with the entity".

In this definition, Mr De Messan emphasizes that true understanding is independent of the belief of the person who understands. It is an absolute function of the logic applied to observations and the accuracy of one’s interpretation of the facts that concern the entity. True understanding can only be achieved if the observer obtains and processes enough information about the entity under scrutiny to arrive at conclusions that will always remain true.

One’s understanding of something only becomes equal to real understanding when the assertions made about the thing are absolute and always true. If future improvements in our logics, formulas, reasoning etc… could invalidate one’s current explanation of a phenomenon, then the understanding that yielded the explanation does not qualify as true understanding, or better still, is only as true as current knowledge permits.

Hence, he argues, there are degrees of understanding as well as understanding at a point in time (a collective state of opinion) all of which ultimately seek to become true understanding.

The conclusion of this definition is that our closeness to true understanding is only as good as the information we possess, the logic of analysis we apply to the information we have and the finality of our reference point.

As this shows, it is difficult to define understanding. If we use the term concept as above, the question then arises as to what is a concept? Is it an abstract thing? Is it a brain pattern or a rule? Whatever definition is proposed, we can still ask how it is that we understand the thing that is featured in the definition: we can never satisfactorily define a concept, still less use it to explain understanding. It may be more convenient to use an operational or behavioral definition, that is, to say that "somebody who reacts appropriately to x understands x". For example, one understands Swahili if one correctly obeys commands given in that language. This approach, however, may not provide an adequate definition. A computer can easily be programmed to react appropriately to commands, but there is a disagreement as to whether or not the computer understands the language (see the Chinese room argument).

According to the independent socionics researcher Rostislav Persion:

In the cognitive model presented by MBTI, the process of introverted thinking (Ti) is thought to represent understanding through cause and effect relationships or correlations. One can construct a model of a system by observing correlations between all the relevant properties (e.g. The output of a NAND gate relative to its inputs). This allows the person to generate truths about the system and then to apply the model to demonstrate his or her understanding. A mechanic for example may randomly, or algorithmically probe the inputs and outputs of a black box to understand the internal components through the use of induction. INTP, ISTP, ESTP, and ENTP all use Ti and are usually the best of the 16 types at understanding their material environment in a bottom-up manner. These types may enjoy mechanics and digital electronics because of the 1 to 1 correlation between cause and effect relationships in these fields. Understanding is not limited to these types however as other types demonstrate an identical process, although in other planes of reality; ie. Social, Theological and Aesthetic. A potential reason for the association of understanding with the former personality types is due to a social phenomenon for asymmetrical distribution of gratification. In the field of engineering, engineers probe or study the inputs and outputs of components to understand their functionality. These components are then combined based on their functionality (similar to computer programming) to create a larger, more complex system. This is the reason why engineers attempt to subdivide ideas as deep as possible to obtain the lowest level of knowledge. This makes their models more detailed and flexible. It may be useful to know the formulas that govern an ideal gas, but to visualise the gas as being made up of small moving particles, which are in turn made up of even smaller particles, is true understanding. People who are understanding (through the use of Ti) usually value objects and people based on usefulness, as opposed to the people who use extroverted thinking (Te) who view people or things as having a worth. In

order to test one's understanding it is necessary to present a question that forces the individual to demonstrate the possession of a model, derived from observable examples of that model's production or potential production (in the case that such a model did not exist beforehand). Rote memorization can present an illusion of understanding, however when other questions are presented with modified attributes within the query, the individual cannot create a solution due to a lack of a deeper representation of reality.

Another significant point of view holds that knowledge is the simple awareness of bits of information. Understanding is the awareness of the connection between the individual pieces of this information. It is understanding which allows knowledge to be put to use. Therefore, understanding represents a deeper level than simple knowledge.

It would be difficult if not impossible to provide citation for the previous point as it is incorrect. The actual order can best be clarified with a common usage example that can be applied to any task, in this case baking a cake. If one is explaining how to bake a cake to someone who has no previous constructs as to what exactly goes into baking a cake and the second person is able to grasp the material presented, it 'makes sense.' This statement that it makes sense does not imply any retention of the information beyond the immediate. If one watches cooking shows and can explain unprompted without immediate previous explanation what is necessary to bake a cake, they 'understand' baking a cake.

This does not imply they have the necessary skill set to actually bake a cake. If one states they have knowledge as in 'I know how to bake a cake' it carries with it the understanding they can take raw ingredients and produce edible results. This example could extend to any common usage I can think of, such as watching construction shows, art shows, etc... may give you understanding of building a house, painting a picture but they do not carry with it the physical application that properly used, saying one 'knows how' does. I believe the previous point was drawn from revised Bloom's Taxonomy in which the original taxonomy had knowledge as the first tier and the revised taxonomy had understanding listed as the second tier. Note that neither model had both terms used together. Furthermore said correlation is in direct contradiction to common usage that has been accepted long before the models so 'if' the models are trying to say this, they are in fact flawed.

One final clarifying point is that of scope. Say we have someone who has watched many shows on deck building, cake making, whatever. That person could have significant understanding that goes beyond the level of a handyman who builds decks in a limited set of variations. The handyman 'knows' how to build a deck but the first person 'understands' a wider scope potentially of deck building. In the final analysis, the difference between understanding and knowledge in the most widely accepted common usage is therefore a matter of application. In reference to Blooms Taxonomy it is a theoretical model that doesn't accurately map common usage so I would submit therefore could use

further revision. Knowledge in common usage would be roughly synonymous with 'Application' in the model.

Gregory Chaitin, a noted computer scientist, propounds a view that comprehension is a kind of data compression. In his essay "The Limits of Reason", he argues that understanding something means being able to figure out a simple set of rules that explains it. For example, we understand why day and night exist because we have a simple model—the rotation of the earth—that explains a tremendous amount of data—changes in brightness, temperature, and atmospheric composition of the earth. We have compressed a large amount of information by using a simple model that predicts it. Similarly, we understand the number 0.33333... by thinking of it as one-third. The first way of representing the number requires an infinite amount of memory; but the second way can produce all the data of the first representation, but uses much less information. Chaitin argues that comprehension is this ability to compress data.

Understanding Basic English

To test Conceptive C, I have decided to write a Basic English Bot that understands basic English enough to have a simple conversation with.

We need a parser that takes input from a Terminal Portal. Parses the input into Words and Sentences, then uses rules to answer the question that is asked and send the output back to the terminal Portal.

We need to build a Dictionary with all of the Basic English 850 Words in it.

Basic English (British American Scientific International Commercial) is a constructed (made-up) language to explain complex thoughts with 850 basic English words chosen by Charles Kay Ogden.

Rules of word use

The word use of Basic English is much simpler and more regular than the word use of full English. Not all the meanings of each word are allowed.

Ogden's rules of grammar for Basic English help people use the 850 words to talk about things and events normally.

-s / -es / -ies change singular nouns into plural nouns.

-ing / -ed change verbs into adjectives.

-ing / -er change verbs into nouns.

-ly change adjectives into adverbs.

-er / -est or more / most describe amounts.

un- change the meanings of adjectives into their opposites.

The opposite word order with do makes questions.

Operators and pronouns conjugate as in normal English.

Make combined words (compounds) from two nouns (for example "milkman") or a noun and a direction (sundown).

Measures, numbers, money, days, months, years, clock time, and international words are in English forms. E.g. Date/Time: 20 May 1972 at 21:00

Use the words of an industry or science. For example, in this grammar, some special words are for teaching languages, and not part of Basic English: plural, conjugate, noun, adjective, adverb, qualifier, operator, pronoun, and directive.

Operations - 100 words

These are the 100 most common words in use in English.

come, get, give, go, keep, let, make, put, seem, take, be, do, have, say, see, send, may, will, about, across, after, against, among, at, before, between, by, down, from, in, off, on, over, through, to, under, up, with, as, for, of, till, than, a, the, all, any, every, no, other, some, such, that, this, I, he, you, who, and, because, but, or, if, though, while, how, when, where, why, again, ever, far, forward, here, near, now, out, still, then, there, together, well, almost, enough, even, little, much, not, only, quite, so, very, tomorrow, yesterday, north, south, east, west, please, yes.

400 General Words

After the most common we have the next 400 words in general use in English.

A-F

account, act, addition, adjustment, advertisement, agreement, air, amount, amusement, animal, answer, apparatus, approval, argument, art, attack, attempt, attention, attraction, authority, back, balance, base, behaviour, belief, birth, bit, bite, blood, blow, body, brass, bread, breath, brother, building, burn, burst, business, butter, canvas, care, cause, chalk, chance, change, cloth, coal, colour, comfort, committee, company, comparison, competition, condition, connection, control, cook, copper, copy, cork, cotton, cough, country, cover, crack, credit, crime, crush, cry, current, curve, damage, danger, daughter, day, death, debt, decision, degree, design, desire, destruction, detail, development, digestion, direction, discovery, discussion, disease, disgust, distance, distribution, division, doubt, drink, driving, dust, earth, edge, education, effect, end, error, event, example, exchange, existence, expansion, experience, expert, fact, fall, family, father, fear, feeling, fiction, field, fight, fire, flame, flight, flower, fold, food, force, form, friend, front, fruit

G-O

glass, gold, government, grain, grass, grip, group, growth, guide, harbour, harmony, hate, hearing, heat, help, history, hole, hope, hour,

humour, ice, idea, impulse, increase, industry, ink, insect, instrument, insurance, interest, invention, iron, jelly, join, journey, judge, jump, kick, kiss, knowledge, land, language, laugh, law, lead, learning, leather, letter, level, lift, light, limit, linen, liquid, list, look, loss, love, machine, man, manager, mark, market, mass, meal, measure, meat, meeting, memory, metal, middle, milk, mind, mine, minute, mist, money, month, morning, mother, motion, mountain, move, music, name, nation, need, news, night, noise, note, number, observation, offer, oil, operation, opinion, order, organization, ornament, owner

P-Z

page, pain, paint, paper, part, paste, payment, peace, person, place, plant, play, pleasure, point, poison, polish, porter, position, powder, power, price, print, process, produce, profit, property, prose, protest, pull, punishment, purpose, push, quality, question, rain, range, rate, ray, reaction, reading, reason, record, regret, relation, religion, representative, request, respect, rest, reward, rhythm, rice, river, road, roll, room, rub, rule, run, salt, sand, scale, science, sea, seat, secretary, selection, self, sense, servant, sex, shade, shake, shame, shock, side, sign, silk, silver, sister, size, sky, sleep, slip, slope, smash, smell, smile, smoke, sneeze, snow, soap, society, son, song, sort, sound, soup, space, stage, start, statement, steam, steel, step, stitch, stone, stop, story, stretch, structure, substance, sugar, suggestion, summer, support, surprise, swim, system, talk, taste, tax, teaching, tendency, test, theory, thing, thought, thunder, time, tin, top, touch, trade, transport, trick, trouble, turn, twist, unit, use, value, verse, vessel, view, voice, walk, war, wash, waste, water, wave, wax, way, weather, week, weight, wind, wine, winter, woman, wood, wool, word, work, wound, writing, year.

Things - 200 Picture-able Words

These are words where you can think of a picture to go with them. In a similar way that a child learns words by looking at pictures with the words underneath.

angle, ant, apple, arch, arm, army, baby, bag, ball, band, basin, basket, bath, bed, bee, bell, berry, bird, blade, board, boat, bone, book, boot, bottle, box, boy, brain, brake, branch, brick, bridge, brush, bucket, bulb, button, cake, camera, card, cart, carriage, cat, chain, cheese, chest, chin, church, circle, clock, cloud, coat, collar, comb, cord, cow, cup, curtain, cushion, dog, door, drain, drawer, dress, drop, ear, egg, engine, eye, face, farm, feather, finger, fish, flag, floor, fly, foot, fork, fowl, frame, garden, girl, glove, goat, gun, hair, hammer, hand, hat, head, heart, hook, horn, horse, hospital, house, island, jewel, kettle, key, knee, knife, knot, leaf, leg, library, line, lip, lock, map, match, monkey, moon, mouth, muscle,

nail, neck, needle, nerve, net, nose, nut, office, orange, oven, parcel, pen, pencil, picture, pig, pin, pipe, plane, plate, plough, pocket, pot, potato, prison, pump, rail, rat, receipt, ring, rod, roof, root, sail, school, scissors, screw, seed, sheep, shelf, ship, shirt, shoe, skin, skirt, snake, sock, spade, sponge, spoon, spring, square, stamp, star, station, stem, stick, stocking, stomach, store, street, sun, table, tail, thread, throat, thumb, ticket, toe, tongue, tooth, town, train, tray, tree, trousers, umbrella, wall, watch, wheel, whip, whistle, window, wing, wire, worm.

Qualities - 100 Descriptive words

These are 100 Words that add a description to other words. For example “A Black Ball.”

able, acid, angry, automatic, beautiful, black, boiling, bright, broken, brown, cheap, chemical, chief, clean, clear, common, complex, conscious, cut, deep, dependent, early, elastic, electric, equal, fat, fertile, first, fixed, flat, free, frequent, full, general, good, great, grey, hanging, happy, hard, healthy, high, hollow, important, kind, like, living, long, male, married, material, medical, military, natural, necessary, new, normal, open, parallel, past, physical, political, poor, possible, present, private, probable, quick, quiet, ready, red, regular, responsible, right, round, same, second, separate, serious, sharp, smooth, sticky, stiff, straight, strong, sudden, sweet, tall, thick, tight, tired, true, violent, waiting, warm, wet, wide, wise, yellow, young.

Qualities - 50 opposites

These words are opposite to other words.

awake, bad, bent, bitter, blue, certain, cold, complete, cruel, dark, dead, dear, delicate, different, dirty, dry, false, feeble, female, foolish, future, green, ill, last, late, left, loose, loud, low, mixed, narrow, old, opposite, public, rough, sad, safe, secret, short, shut, simple, slow, small, soft, solid, special, strange, thin, white, wrong.

Consciousness

Consciousness is a term that refers to the relationship between the mind and the world with which it interacts. It has been defined as: subjectivity, awareness, the ability to experience or to feel, wakefulness, having a sense of selfhood, and the executive control system of the mind. Despite the difficulty in definition, many philosophers believe that there is a broadly shared underlying intuition about what consciousness is. As Max Velmans and Susan Schneider wrote in The Blackwell Companion to Consciousness: "Anything that we are aware of at a given moment forms part of our consciousness, making conscious experience at once the most familiar and most mysterious aspect of our lives."

Philosophers since the time of Descartes and Locke have struggled to comprehend the nature of consciousness and pin down its essential properties. Issues of concern in the philosophy of consciousness include whether the concept is fundamentally valid; whether consciousness can ever be explained mechanistically; whether non-human consciousness exists and if so how it can be recognized; how consciousness relates to language; and whether it may ever be possible for computers or robots to be conscious. Perhaps the thorniest issue is whether consciousness can be understood in a way that does not require a dualistic distinction between mental and physical states or properties.

At one time consciousness was viewed with skepticism by many scientists, but in recent years it has become a significant topic of research in psychology and neuroscience. The primary focus is on understanding what it means biologically and psychologically for information to be present in consciousness —that is, on determining the neural and psychological correlates of consciousness. The majority of experimental studies assess consciousness by asking human subjects for a verbal report of their experiences (e.g., "tell me if you notice anything when I do this"). Issues of interest include phenomena such as subliminal perception, blindsight, denial of impairment, and altered states of consciousness produced by psychoactive drugs or spiritual or meditative techniques.

In medicine, consciousness is assessed by observing a patient's arousal and responsiveness, and can be seen as a continuum of states ranging from full alertness and comprehension, through disorientation, delirium, loss of meaningful communication, and finally loss of movement in response to painful stimuli. Issues of practical concern include how the presence of consciousness can be assessed in severely ill, comatose, or anesthetized people, and how to treat conditions in which consciousness is impaired or disrupted.

Etymology and Early History

The origin of the modern concept of consciousness is often attributed to John Locke's Essay Concerning Human Understanding, published in 1690. Locke defined consciousness as “the perception of what passes in a man’s own mind.”

His essay influenced the 18th century view of consciousness, and his definition appeared in Samuel Johnson's celebrated Dictionary (1755).

The earliest English language uses of "conscious" and "consciousness" date back, however, to the 1500s. The English word "conscious" originally derived from the Latin conscius (con- "together" + scire "to know"), but the Latin word did not have the same meaning as our word—it meant knowing with, in other words having joint or common knowledge with another. There were, however, many occurrences in Latin writings of the phrase conscius sibi, which translates literally as "knowing with oneself", or in other words sharing knowledge with oneself about something. This phrase had the figurative meaning of knowing that one knows, as the modern English word "conscious" does. In its earliest uses in the 1500s, the English word "conscious" retained the meaning of the Latin conscius. For example Thomas Hobbes in Leviathan wrote: "Where two, or more men, know of one and the same fact, they are said to be Conscious of it one to another." The Latin phrase conscius sibi, whose meaning was more closely related to the current concept of consciousness, was rendered in English as "conscious to oneself" or "conscious unto oneself". For example, Archbishop Ussher wrote in 1613 of "being so conscious unto myself of my great weakness". Locke's definition from 1690 illustrates that a gradual shift in meaning had taken place.

A related word was conscientia, which primarily means moral conscience. In the literal sense, "conscientia" means knowledge-with, that is, shared knowledge. The word first appears in Latin juridical texts by writers such as Cicero. Here, conscientia is the knowledge that a witness has of the deed of someone else. René Descartes (1596–1650) is generally taken to be the first philosopher to use "conscientia" in a way that does not fit this traditional meaning. Descartes used "conscientia" the way modern speakers would use "conscience." In Search after Truth he says "conscience or internal testimony" (conscientia vel interno testimonio).

In Philosophy

The philosophy of mind has given rise to many stances regarding consciousness. Any attempt to impose an organization on them is bound to be somewhat arbitrary. Stuart Sutherland exemplified the difficulty in the entry he wrote for the 1989 version of the Macmillan Dictionary of Psychology:

Consciousness—The having of perceptions, thoughts, and feelings; awareness. The term is impossible to define except in terms that are unintelligible without a grasp of what consciousness means. Many fall into the trap of equating consciousness with self-consciousness—to be conscious it is only necessary to be aware of the external world. Consciousness is a fascinating but elusive phenomenon: it is impossible to specify what it is, what it does, or why it has evolved. Nothing worth reading has been written on it.

Most writers on the philosophy of consciousness have been concerned to defend a particular point of view, and have organized their material

accordingly. For surveys, the most common approach is to follow a historical path by associating stances with the philosophers who are most most strongly associated with them, for example Descartes, Locke, Kant, etc. The main alternative, followed in the present article, is to organize philosophical stances according to the answers they give to a set of basic questions about the nature and status of consciousness.

Is Consciousness a Valid Concept?

The most compelling argument for the existence of consciousness is that the vast majority of mankind have an overwhelming intuition that there truly is such a thing. Skeptics argue that this intuition, in spite of its compelling quality, is false, either because the concept of consciousness is intrinsically incoherent, or because our intuitions about it are based in illusions. Gilbert Ryle, for example, argued that traditional understanding of consciousness depends on a Cartesian dualist outlook that improperly distinguishes between mind and body, or between mind and world. He proposed that we speak not of minds, bodies, and the world, but of individuals, or persons, acting in the world. Thus, by speaking of 'consciousness' we end up misleading ourselves by thinking that there is any sort of thing as consciousness separated from behavioral and linguistic understandings. More generally, many philosophers and scientists have been unhappy about the difficulty of producing a definition that does not involve circularity or fuzziness.

Is It a Single Thing?

Many philosophers have argued that consciousness is a unitary concept that is understood intuitively by the majority of people in spite of the difficulty in defining it. Others, though, have argued that the level of disagreement about the meaning of the word indicates that it either means different things to different people, or else is an umbrella term encompassing a variety of distinct meanings with no simple element in common.

Ned Block proposed a distinction between two types of consciousness that he called phenomenal (P-consciousness) and access (A-consciousness). P-consciousness, according to Block, is simply raw experience: it is moving, colored forms, sounds, sensations, emotions and feelings with our bodies and responses at the center. These experiences, considered independently of any impact on behavior, are called qualia. A-consciousness, on the other hand, is the phenomenon whereby information in our minds is accessible for verbal report, reasoning, and the control of behavior. So, when we perceive, information about what we perceive is access conscious; when we introspect, information about our thoughts is access conscious; when we remember, information about the past is access conscious, and so on. Although some philosophers, such as Daniel Dennett, have disputed the validity of this distinction, others have broadly accepted it. David Chalmers has argued that A-consciousness can in principle be understood in mechanistic terms, but that

understanding P-consciousness is much more challenging: he calls this the hard problem of consciousness.

Some philosophers believe that Block's two types of consciousness are not the end of the story. William Lycan, for example, argued in his book Consciousness and Experience that at least eight clearly distinct types of consciousness can be identified (organism consciousness; control consciousness; consciousness of; state/event consciousness; reportability; introspective consciousness; subjective consciousness; self-consciousness)—and that even this list omits several more obscure forms.

How does it relate to the Physical World?

Illustration of dualism by René Descartes. Inputs are passed by the sensory organs to the pineal gland and from there to the immaterial spirit.

The first influential philosopher to discuss this question specifically was Descartes, and the answer he gave is known as Cartesian dualism. Descartes proposed that consciousness resides within an immaterial domain he called res cogitans (the realm of thought), in contrast to the domain of material things which he called res extensa (the realm of extension). He suggested that the interaction between these two domains occurs inside the brain, perhaps in a small midline structure called the pineal gland.

Although it is widely accepted that Descartes explained the problem cogently, few later philosophers have been happy with his solution, and his ideas about the pineal gland have especially been ridiculed.

Alternative solutions, however, have been very diverse. They can be divided broadly into two categories: dualist solutions that maintain Descartes's rigid distinction between the realm of consciousness and the realm of matter but give different answers for how the two realms relate to each other; and monist solutions that maintain that there is really only one realm of being, of which consciousness and matter are both aspects. Each of these categories itself contains numerous variants. The two main types of dualism are substance dualism (which holds that the mind is formed of a distinct type of substance not governed by the laws of physics) and property dualism (which holds that the laws of physics are universally valid but cannot be used to explain the mind). The three main types of monism are physicalism (which holds that the mind consists of matter organized in a particular way), idealism (which holds that only thought truly exists and matter is merely an illusion), and neutral monism (which holds that both mind and matter are aspects of a distinct essence that is itself identical to neither of them). There are also, however, a large number of idiosyncratic theories that cannot cleanly be assigned to any of these camps.

Since the dawn of Newtonian science with its vision of simple mechanical principles governing the entire universe, some philosophers have been tempted by the idea that consciousness could be explained in purely physical terms. The first influential writer to propose such an idea explicitly was Julien Offray de

La Mettrie, in his book Man a Machine (L'homme machine). His arguments, however, were very abstract. The most influential modern physical theories of consciousness are based on psychology and neuroscience. Theories proposed by neuroscientists such as Gerald Edelman[31] and Antonio Damasio, and by philosophers such as Daniel Dennett,[33] seek to explain consciousness in terms of neural events occurring within the brain. Many other neuroscientists, such as Christof Koch, have explored the neural basis of consciousness without attempting to frame all-encompassing global theories. At the same time, computer scientists working in the field of Artificial Intelligence have pursued the goal of creating digital computer programs that can simulate or embody consciousness.

A few theoretical physicists have argued that classical physics is intrinsically incapable of explaining the holistic aspects of consciousness, but that quantum theory provides the missing ingredients. Several theorists have therefore proposed quantum mind (QM) theories of consciousness. Notable theories falling into this category include the Holonomic brain theory of Kark Pribram and David Bohm, and the Orch-OR theory formulated by Stuart Hameroff and Roger Penrose. Some of these QM theories offer descriptions of phenomenal consciousness, as well as QM interpretations of access consciousness. None of the quantum mechanical theories has been confirmed by experiment, and many scientists and philosophers consider the arguments for an important role of quantum phenomena to be unconvincing.

Why do people believe that other people are Conscious?

Many philosophers consider experience to be the essence of consciousness, and believe that experience can only fully be known from the inside, subjectively. But if consciousness is subjective and not visible from the outside, why do the vast majority of people believe that other people are conscious, but rocks and trees are not? This is called the problem of other minds. It is particularly acute for people who believe in the possibility of philosophical zombies, that is, people who think it is possible in principle to have an entity that is physically indistinguishable from a human being and behaves like a human being in every way but nevertheless lacks consciousness.

The most commonly given answer is that we attribute consciousness to other people because we see that they resemble us in appearance and behavior: we reason that if they look like us and act like us, they must be like us in other ways, including having experiences of the sort that we do.[40] There are, however, a variety of problems with that explanation. For one thing, it seems to violate the principle of parsimony, by postulating an invisible entity that is not necessary to explain what we observe. Some philosophers, such as Daniel Dennett in an essay titled The Unimagined Preposterousness of Zombies, argue that people who give this explanation do not really understand what they are saying.

More broadly, philosophers who do not accept the possibility of zombies generally believe that consciousness is reflected in behavior (including verbal behavior), and that we attribute consciousness on the basis of behavior. A more straightforward way of saying this is that we attribute experiences to people because of what they can do, including the fact that they can tell us about their experiences.

Are non-human animals are Conscious?

The topic of animal consciousness is beset by a number of difficulties. It poses the problem of other minds in an especially severe form, because animals, lacking language, cannot tell us about their experiences. Also, it is difficult to reason objectively about the question, because a denial that an animal is conscious is often taken to imply that it does not feel, its life has no value, and that harming it is not morally wrong. Descartes, for example, has sometimes been blamed for mistreatment of animals due to the fact that he believed only humans have a non-physical mind. Most people have a strong intuition that some animals, such as dogs, are conscious, while others, such as insects, are not; but the sources of this intuition are not obvious.

Philosophers who consider subjective experience the essence of consciousness also generally believe, as a correlate, that the existence and nature of animal consciousness can never rigorously be known. Thomas Nagel spelled out this point of view in an influential essay titled What Is it Like to Be a Bat?. He said that an organism is conscious "if and only if there is something that it is like to be that organism — something it is like for the organism"; and he argued that no matter how much we know about an animal's brain and behavior, we can never really put ourselves into the mind of the animal and experience its world in the way it does itself. Other thinkers, such as Douglas Hofstadter, dismiss this argument as incoherent. Several psychologists and ethologists have argued for the existence of animal consciousness by describing a range of behaviors that appear to show animals holding beliefs about things they cannot directly perceive — Donald Griffin's 2001 book Animal Minds reviews a substantial portion of the evidence.

Could a Machine ever be Conscious?

The idea of an artifact made conscious is an ancient theme of mythology, appearing for example in the Greek myth of Pygmalion, who carved a statue that was magically brought to life, and in medieval Jewish stories of the Golem, a magically animated homunculus built of clay. However, the possibility of actually constructing a conscious machine was probably first discussed by Ada Lovelace, in a set of notes written in 1842 about the Analytical Engine invented by Charles Babbage, a precursor (never built) to modern electronic computers. Lovelace was essentially dismissive of the idea that a machine such as the Analytical Engine could think in a humanlike way. She wrote:

It is desirable to guard against the possibility of exaggerated ideas that might arise as to the powers of the Analytical Engine. ... The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform. It can follow analysis; but it has no power of anticipating any analytical relations or truths. Its province is to assist us in making available what we are already acquainted with.

One of the most influential contributions to this question was an essay written in 1950 by pioneering computer scientist Alan Turing, titled Computing Machinery and Intelligence. Turing disavowed any interest in terminology, saying that even "Can machines think?" is too loaded with spurious connotations to be meaningful; but he proposed to replace all such questions with a specific operational test, which has become known as the Turing test. To pass the test a computer must be able to imitate a human well enough to fool interrogators. In his essay Turing discussed a variety of possible objections, and presented a counterargument to each of them. The Turing test is commonly cited in discussions of artificial intelligence as a proposed criterion for machine consciousness; it has provoked a great deal of philosophical debate. For example, Daniel Dennett and Douglas Hofstadter argue that anything capable of passing the Turing test is necessarily conscious, while David Chalmers argues that a philosophical zombie could pass the test, yet fail to be conscious.

Some philosophers have argued that it might be possible, at least in principle, for a machine to be conscious, but that this would not just be a matter of executing the right computer program. This viewpoint has most notably been advocated by John Searle, who defended it using a thought experiment that has become known as the Chinese room argument: Suppose the Turing test is conducted in Chinese rather than English, and suppose a computer program successfully passes it. Does the system that is executing the program understand Chinese? Searle's argument is that he could pass the Turing test in Chinese too, by executing exactly the same program as the computer, yet without understanding a word of Chinese. How does Searle know whether he understands Chinese? Understanding a language is not just something you do: it feels like something to understand Chinese (or any language). Understanding is conscious, and therefore there is more to it than merely executing a program, even a Turing-test-passing program.

In the literature concerning artificial consciousness, Searle's essay has been second only to Turing's in the volume of debate it has generated. Searle himself was vague about what extra ingredients it would take to make a machine conscious: all he proposed was that what was needed was "causal powers" of the sort that the brain has and that computers lack. But other thinkers sympathetic to his basic argument have suggested that the necessary (though perhaps still not sufficient) extra conditions may include the ability to pass not just the verbal version of the Turing test, but the robotic version,[54] which requires grounding the robot's words in the robot's sensorimotor capacity to categorize and interact with the things in the world that its words are about,

Turing-indistinguishably from a real person. Turing-scale robotics is an empirical branch of research on embodied cognition and situated cognition.

Spiritual Approaches

To most philosophers, the word "consciousness" connotes the relationship between the mind and the world. To writers on spiritual or religious topics, it frequently connotes the relationship between the mind and God, or the relationship between the mind and deeper truths that are thought to be more fundamental than the physical world. Krishna consciousness, for example, is a term used to mean an intimate linkage between the mind of a worshipper and the god Krishna. The mystical psychiatrist Richard Maurice Bucke distinguished between three types of consciousness: Simple Consciousness, awareness of the body, possessed by many animals; Self Consciousness, awareness of being aware, possessed only by humans; and Cosmic Consciousness, awareness of the life and order of the universe, possessed only by humans who are enlightened. Many more examples could be given. The most thorough account of the spiritual approach may be Ken Wilber's book The Spectrum of Consciousness, a comparison of western and eastern ways of thinking about the mind. Wilber described consciousness as a spectrum with ordinary awareness at one end, and more profound types of awareness at higher levels.

Scientific Approaches

For many decades, consciousness as a research topic was avoided by the majority of mainstream scientists, because of a general feeling that a phenomenon defined in subjective terms could not properly be studied using objective experimental methods. Starting in the 1980s, though, an expanding community of neuroscientists and psychologists have associated themselves with a field called Consciousness Studies, giving rise to a stream of experimental work published in journals such as Consciousness and Cognition, and methodological work published in journals such as the Journal of Consciousness Studies, along with regular conferences organized by groups such as the Association for the Scientific Study of Consciousness.

Modern scientific investigations into consciousness are based on psychological experiments (including, for example, the investigation of priming effects using subliminal stimuli), and on case studies of alterations in consciousness produced by trauma, illness, or drugs. Broadly viewed, scientific approaches are based on two core concepts. The first identifies the content of consciousness with the experiences that are reported by human subjects; the second makes use of the concept of consciousness that has been developed by neurologists and other medical professionals who deal with patients whose behavior is impaired. In either case, the ultimate goals are to develop techniques for assessing consciousness objectively in humans as well as other animals, and to understand the neural and psychological mechanisms that underlie it.

Biological Function and Evolution

Regarding the primary function of conscious processing, a recurring idea in recent theories is that phenomenal states somehow integrate neural activities and information-processing that would otherwise be independent. This has been called the integration consensus. However, it remains unspecified which kinds of information are integrated in a conscious manner and which kinds can be integrated without consciousness. Nor is it explained what specific causal role conscious integration plays, nor why the same functionality cannot be achieved without consciousness. Obviously not all kinds of information are capable of being disseminated consciously (e.g., neural activity related to vegetative functions, reflexes, unconscious motor programs, low-level perceptual analyses, etc.) and many kinds of information can be disseminated and combined with other kinds without consciousness, as in intersensory interactions such as the ventriloquism effect. Hence it remains unclear why any of it is conscious.

As noted earlier, even among writers who consider consciousness to be a well-defined thing, there is widespread dispute about which animals other than humans can be said to possess it. Thus, any examination of the evolution of consciousness is faced with great difficulties. Nevertheless, some writers have argued that consciousness can be viewed from the standpoint of evolutionary biology as an adaptation in the sense of a trait that increases fitness. In his paper "Evolution of consciousness," John Eccles argued that special anatomical and physical properties of the mammalian cerebral cortex gave rise to consciousness. Bernard Baars proposed that once in place, this "recursive" circuitry may have provided a basis for the subsequent development of many of the functions that consciousness facilitates in higher organisms.

Other philosophers, however, have suggested that consciousness would not be necessary for any functional advantage in evolutionary processes. No one has given a causal explanation, they argue, of why it would not be possible for a functionally equivalent non-conscious organism (i.e., a philosophical zombie) to achieve the very same survival advantages as a conscious organism. If evolutionary processes are blind to the difference between function F being performed by conscious organism O and non-conscious organism O*, it is unclear what adaptive advantage consciousness could provide.

States of Consciousness

There are some states in which consciousness seems to be abolished, including sleep, coma, and death. There are also a variety of circumstances that can change the relationship between the mind and the world in less drastic ways, producing what are known as altered states of consciousness. Some altered states occur naturally; others can be produced by drugs or brain damage.

The two most widely accepted altered states are sleep and dreaming. Although dream sleep and non-dream sleep appear very similar to an outside observer, each is associated with a distinct pattern of brain activity, metabolic activity,

and eye movement; each is also associated with a distinct pattern of experience and cognition. During ordinary non-dream sleep, people who are awakened report only vague and sketchy thoughts, and their experiences do not cohere into a continuous narrative. During dream sleep, in contrast, people who are awakened report rich and detailed experiences in which events form a continuous progression, which may however be interrupted by bizarre or fantastic intrusions. Thought processes during the dream state frequently show a high level of irrationality. Both dream and non-dream states are associated with severe disruption of memory: it usually disappears in seconds during the non-dream state, and in minutes after awakening from a dream unless actively refreshed.

A variety of psychoactive drugs have notable effects on consciousness. These range from a simple dulling of awareness produced by sedatives, to increases in the intensity of sensory qualities produced by stimulants, cannabis, or most notably by the class of drugs known as psychedelics. LSD, mescaline, psilocybin, and others in this group can produce major distortions of perception, including hallucinations; some users even describe their drug-induced experiences as mystical or spiritual in quality. The brain mechanisms underlying these effects are not well understood, but there is substantial evidence that alterations in the brain system that uses the chemical neurotransmitter serotonin play an essential role.

There has been some research into physiological changes in yogis and people who practise various techniques of meditation. Some research with brain waves during meditation has differences between those corresponding to ordinary relaxation and those corresponding to meditation. It has been disputed, however, whether there is enough evidence to count these as physiologically distinct states of consciousness.

The most extensive study of the characteristics of altered states of consciousness was made by psychologist Charles Tart in the 1960s and 1970s. Tart analyzed a state of consciousness as made up of a number of component processes, including exteroception (sensing the external world); interoception (sensing the body); input-processing (seeing meaning); emotions; memory; time sense; sense of identity; evaluation and cognitive processing; motor output; and interaction with the environment. Each of these, in his view, could be altered in multiple ways by drugs or other manipulations. The components that Tart identified have not, however, been validated by empirical studies. Research in this area has not yet reached firm conclusions, but a recent questionnaire-based study identified eleven significant factors contributing to drug-induced states of consciousness: experience of unity; spiritual experience; blissful state; insightfulness; disembodiment; impaired control and cognition; anxiety; complex imagery; elementary imagery; audio-visual synesthesia; and changed meaning of percepts.

Phenomenology

Phenomenology is a method of inquiry that attempts to examine the structure of consciousness in its own right, putting aside problems regarding the relationship of consciousness to the physical world. This approach was first proposed by the philosopher Edmund Husserl, and later elaborated by other philosophers and scientists. Husserl's original concept gave rise to two distinct lines of inquiry, in philosophy and psychology. In philosophy, phenomenology has largely been devoted to fundamental metaphysical questions, such as the nature of intentionality ("aboutness"). In psychology, phenomenology largely has meant attempting to investigate consciousness using the method of introspection, which means looking into one's own mind and reporting what one observes. This method fell into disrepute in the early twentieth century because of grave doubts about its reliability, but has been rehabilitated to some degree, especially when used in combination with techniques for examining brain activity.

Introspectively, the world of conscious experience seems to have considerable structure. Immanuel Kant asserted that the world as we perceive it is organized according to a set of fundamental "intuitions", which include object (we perceive the world as a set of distinct things); shape; quality (color, warmth, etc.); space (distance, direction, and location); and time. Some of these constructs, such as space and time, correspond to the way the world is structured by the laws of physics; for others the correspondence is not as clear. Understanding the physical basis of qualities, such as redness or pain, has been particularly challenging. David Chalmers has called this the hard problem of consciousness. Some philosophers have argued that it is intrinsically unsolvable, because qualities ("qualia") are ineffable; that is, they are "raw feels", incapable of being analyzed into component processes. Most psychologists and neuroscientists have not accepted these arguments — nevertheless it is clear that the relationship between a physical entity such as light and a perceptual quality such as color is extraordinarily complex and indirect, as demonstrated by a variety of optical illusions such as neon color spreading.

In neuroscience, a great deal of effort has gone into investigating how the perceived world of conscious awareness is constructed inside the brain. The process is generally thought to involve two primary mechanisms: (1) hierarchical processing of sensory inputs, (2) memory. Signals arising from sensory organs are transmitted to the brain and then processed in a series of stages, which extract multiple types of information from the raw input. In the visual system, for example, sensory signals from the eyes are transmitted to the thalamus and then to the primary visual cortex; inside the cerebral cortex they are sent to areas that extract features such as three-dimensional structure, shape, color, and motion. Memory comes into play in at least two ways. First, it allows sensory information to be evaluated in the context of previous experience. Second, and even more importantly, working memory allows information to be integrated over time so that it can generate a stable

representation of the world—Gerald Edelman expressed this point vividly by titling one of his books about consciousness The Remembered Present.

Despite the large amount of information available, the most important aspects of perception remain mysterious. A great deal is known about low-level signal processing in sensory systems, but the ways by which sensory systems interact with each other, with "executive" systems in the frontal cortex, and with the language system are very incompletely understood. At a deeper level, there are still basic conceptual issues that remain unresolved. Many scientists have found it difficult to reconcile the fact that information is distributed across multiple brain areas with the apparent unity of consciousness: this is one aspect of the so-called binding problem. There are also some scientists who have expressed grave reservations about the idea that the brain forms representations of the outside world at all: influential members of this group include psychologist J. J. Gibson and roboticist Rodney Brooks, who both argued in favor of "intelligence without representation".

Assessment

In medicine, consciousness is examined using a set of procedures known as neuropsychological assessment. There are two commonly used methods for assessing the level of consciousness of a patient: a simple procedure that requires minimal training, and a more complex procedure that requires substantial expertise. The simple procedure begins by asking whether the patient is able to move and react to physical stimuli. If so, the next question is whether the patient can respond in a meaningful way to questions and commands. If so, the patient is asked for name, current location, and current day and time. A patient who can answer all of these questions is said to be "oriented times three" (sometimes denoted "Ox3" on a medical chart), and is usually considered fully conscious.

The more complex procedure is known as a neurological examination, and is usually carried out by a neurologist in a hospital setting. A formal neurological examination runs through a precisely delineated series of tests, beginning with tests for basic sensorimotor reflexes, and culminating with tests for sophisticated use of language. The outcome may be summarized using the Glasgow Coma Scale, which yields a number in the range 3—15, with a score of 3 indicating brain death (the lowest defined level of consciousness), and 15 indicating full consciousness. The Glasgow Coma Scale has three subscales, measuring the best motor response (ranging from "no motor response" to "obeys commands"), the best eye response (ranging from "no eye opening" to "eyes opening spontaneously") and the best verbal response (ranging from "no verbal response" to "fully oriented"). There is also a simpler pediatric version of the scale, for children too young to be able to use language.

The Stream of Consciousness

William James is usually credited with popularizing the idea that human consciousness flows like a stream, in his Principles of Psychology of 1890.

According to James, the "stream of thought" is governed by five characteristics: "(1) Every thought tends to be part of a personal consciousness. (2) Within each personal consciousness thought is always changing. (3) Within each personal consciousness thought is sensibly continuous. (4) It always appears to deal with objects independent of itself. (5) It is interested in some parts of these objects to the exclusion of others". A similar concept appears in Buddhist philosophy, expressed by the Sanskrit term Citta-sa□ tāna, which is usually translated as mindstream or "mental continuum". In the Buddhist view, though, the "mindstream" is viewed primarily as a source of noise that distracts attention from a changeless underlying reality.

In the west, the primary impact of the idea has been on literature rather than science: stream of consciousness as a narrative mode means writing in a way that attempts to portray the moment-to-moment thoughts and experiences of a character. This technique perhaps had its beginnings in the monologues of Shakespeare's plays, and reached its fullest development in the novels of James Joyce and Virginia Woolf, although it has also been used by many other noted writers.

Thought Experiment

This has nothing to do with Conceptive C, but it is an interesting AI problem that will make you think about how far we can go with an AI Computer Language. As far as Chinese goes Google does a very good job of translating Chinese to English and back again. This does not imply that Google is a conscious AI computer program.

I fall into the “Why the hell should I learn Chinese just to do your thought experiment” but I think you can program a computer to pass the Turing Test for AI. If you you started from Basic English vocabulary and then extended the model to handle 10,000 Words. I would think you would start getting fairly intelligent conversations. Continuing on to 20,000 or 30,000 Words could be done, once you had a working model. I don’t think the computer would be conscious, as far as understanding goes, I would judge it’s level of understanding by the quality of the replies to questions.

The Chinese Room

If you can carry on an intelligent conversation using pieces of paper slid under a door, does this imply that someone or something on the other side understands what you are saying?

The Chinese Room is a thought experiment by John Searle which first appeared in his paper "Minds, Brains, and Programs", published in Behavioral and Brain Sciences in 1980. It addresses the question: if a machine can convincingly simulate an intelligent conversation, does it necessarily understand? In the experiment Searle imagines himself in a room, acting as a computer by manually executing a program that convincingly simulates the behavior of a native Chinese speaker. People outside the room slide Chinese characters under the door and Searle, to whom "Chinese writing is just so many meaningless squiggles", is able to create sensible replies, in Chinese, by following the instructions of the program; that is, by moving papers around. The question arises whether Searle can be said to understand Chinese in the same way that, as Searle says:

according to strong AI, . . . the appropriately programmed computer really is a mind, in the sense that computers given the right programs can be literally said to understand and have other cognitive states.

The experiment is the centerpiece of Searle's Chinese Room Argument which holds that a program cannot give a computer a "mind" or "understanding", regardless of how intelligently it may make it behave. He concludes that "programs are neither constitutive of nor sufficient for minds." "I can have any formal program you like, but I still understand nothing."

The Chinese room is an argument against certain claims of leading thinkers in the field of artificial intelligence, and is not concerned with the level of intelligence that an AI program can display. Searle's argument is directed

against functionalism and computationalism (philosophical positions inspired by AI), rather than the goals of applied AI research itself. The argument leaves aside the question of creating an artificial mind by methods other than symbol manipulation.

Controversial, and the subject of an entire literature of counterargument, it became Behavioral and Brain Sciences's "most influential target article", generating an enormous number of commentaries and responses in the ensuing decades.

Chinese Room Thought Experiment

Searle's thought experiment begins with this hypothetical premise: suppose that artificial intelligence research has succeeded in constructing a computer that behaves as if it understands Chinese. It takes Chinese characters as input and, by following the instructions of a computer program, produces other Chinese characters, which it presents as output. Suppose, says Searle, that this computer performs its task so convincingly that it comfortably passes the Turing test: it convinces a human Chinese speaker that the program is itself a live Chinese speaker. To all of the questions that the person asks, it makes appropriate responses, such that any Chinese speaker would be convinced that he or she is talking to another Chinese-speaking human being.

The question Searle wants to answer is this: does the machine literally "understand" Chinese? Or is it merely simulating the ability to understand Chinese? Searle calls the first position "strong AI" (see below) and the latter "weak AI".

Searle then supposes that he is in a closed room and has a book with an English version of the computer program, along with sufficient paper, pencils, erasers, and filing cabinets. Searle could receive Chinese characters through a slot in the door, process them according to the program's instructions, and produce Chinese characters as output. As the computer had passed the Turing test this way, it is fair, says Searle, to deduce that he would be able to do so as well, simply by running the program manually.

Searle asserts that there is no essential difference between the role the computer plays in the first case and the role he plays in the latter. Each is simply following a program, step-by-step, which simulates intelligent behavior. And yet, Searle points out, "I don't speak a word of Chinese." Since he does not understand Chinese, Searle argues, we must infer that the computer does not understand Chinese either.

Searle argues that without "understanding" (what philosophers call "intentionality"), we cannot describe what the machine is doing as "thinking". Because it does not think, it does not have a "mind" in anything like the normal sense of the word, according to Searle. Therefore, he concludes, "strong AI" is mistaken.

History

Searle's argument first appeared in his paper "Minds, Brains, and Programs", published in Behavioral and Brain Sciences in 1980. It eventually became the journal's "most influential target article", generating an enormous number of commentaries and responses in the ensuing decades.

Most of the discussion consists of attempts to refute it. "The overwhelming majority," notes BBS editor Stevan Harnad, "still think that the Chinese Room Argument is dead wrong." The sheer volume of the literature that has grown up around it inspired Pat Hayes to quip that the field of cognitive science ought to be redefined as "the ongoing research program of showing Searle's Chinese Room Argument to be false."

The paper has become "something of a classic in cognitive science," according to Harnad. Varol Akman agrees, and has described Searle's paper as "an exemplar of philosophical clarity and purity".

Computationalism

Although the Chinese Room argument was originally presented in reaction to the statements of AI researchers, philosophers have come to view it as an important part of the philosophy of mind. It is a challenge to functionalism and the computational theory of mind, and is related to such questions as the mind-body problem, the problem of other minds, the symbol-grounding problem, and the hard problem of consciousness.

Strong AI

Searle identified a philosophical position he calls "strong AI":

The appropriately programmed computer with the right inputs and outputs would thereby have a mind in exactly the same sense human beings have minds.

The definition hinges on the distinction between simulating a mind and actually having a mind. Searle writes that "according to Strong AI, the correct simulation really is a mind. According to Weak AI, the correct simulation is a model of the mind."

The position is implicit in some of the statements of early AI researchers and analysts. For example, in 1955, AI founder Herbert Simon declared that "there are now in the world machines that think, that learn and create" and claimed that they had "solved the venerable mind-body problem, explaining how a system composed of matter can have the properties of mind." John Haugeland wrote that "AI wants only the genuine article: machines with minds, in the full and literal sense. This is not science fiction, but real science, based on a theoretical conception as deep as it is daring: namely, we are, at root, computers ourselves."

Searle also ascribes the following positions to advocates of strong AI:

AI systems can be used to explain the mind;

The study of the brain is irrelevant to the study of the mind; and The Turing test is adequate for establishing the existence of mental states.

Strong AI as Functionalism

In more recent presentations of the Chinese room argument, Searle has identified "strong AI" as "computer functionalism" (a term he attributes to Daniel Dennett). Functionalism is a position in modern philosophy of mind that holds that we can define mental phenomena (such as beliefs, desires, and perceptions) by describing their functions in relation to each other and to the outside world. Because a computer program can accurately represent functional relationships as relationships between symbols, a computer can have mental phenomena if it runs the right program, according to functionalism.

Stevan Harnad argues that Searle's depictions of strong AI can be reformulated as "recognizable tenets of computationalism, a position (unlike 'strong AI') that is actually held by many thinkers, and hence one worth refuting." Computationalism is the position in the philosophy of mind which argues that the mind can be accurately described as an information-processing system.

Each of the following, according to Harnad, is a "tenet" of computationalism:

Mental states are computational states (which is why computers can have mental states and help to explain the mind);

Computational states are implementation-independent — in other words, it is the software that determines the computational state, not the hardware (which is why the brain, being hardware, is irrelevant); and that

Since implementation is unimportant, the only empirical data that matters is how the system functions; hence the Turing test is definitive.

Computers vs. Machines vs. Brains

The Chinese room has exactly the same design as any modern computer. It has a Von Neumann architecture, which consists of a program (the book of instructions), some memory (the papers and file cabinets), a CPU which follows the instructions (the man), and a means to write symbols in memory (the pencil and eraser). A machine with this design is known in theoretical computer science as "Turing complete", because it has the necessary machinery to carry out any computation that a Turing machine can do, and therefore it is capable of doing a step-by-step simulation of any other digital machine. Alan Turing writes, "all digital computers are in a sense equivalent." In other words, the Chinese room can do whatever any other computer can do (albeit much, much more slowly). The widely accepted Church-Turing thesis holds that anything computable is computable by any Turing complete machine.

The Chinese room (and all modern computers) manipulates physical objects in order to carry out calculations and do simulations. AI researchers Allen Newell and Herbert Simon called this kind of machine a physical symbol system. It is

also equivalent to the formal systems used in the field of mathematical logic. Searle emphasizes the fact that this kind of symbol manipulation is syntactic (borrowing a term from the study of grammar). The CPU manipulates the symbols using a form of syntax rules, without any knowledge of the symbol's semantics (that is, their meaning).

Searle's argument applies specifically to computers (that is, devices that can only manipulate symbols without knowing what they mean) and not to machines in general. Searle does not disagree that machines can have consciousness and understanding, because, as he writes, "we are precisely such machines". Searle holds that the brain is, in fact, a machine, but the brain gives rise to consciousness and understanding using machinery that is non-computational. Searle writes "brains cause minds" and that "actual human mental phenomena [are] dependent on actual physical-chemical properties of actual human brains", a position called "biological naturalism" (as opposed to alternatives like dualism, behaviorism, functionalism or identity theory). Indeed, Searle accuses "strong AI" of dualism, the idea that the brain and mind are made of different "substances". He writes that "strong AI only makes sense given the dualistic assumption that, where the mind is concerned, the brain doesn't matter."

Intentionality vs. Consciousness

Searle's original argument centered on 'understanding' — that is, mental states with what philosophers call 'intentionality' — and did not directly address other closely related ideas such as 'consciousness'. David Chalmers argued that "it is fairly clear that consciousness is at the root of the matter". In more recent presentations of the Chinese Room, Searle has included 'consciousness' as part of the argument as well.

Strong AI vs. AI research

Searle's argument does not limit the intelligence with which machines can behave or act; indeed, it does not address this issue directly. "The Chinese room argument ... assumes complete success on the part of artificial intelligence in simulating human cognition," Searle writes. This leaves open the possibility that a machine could be built that acts more intelligent than a person, but does not have a mind or intentionality in the same way that brains do.

Since the primary mission of artificial intelligence research is only to create useful systems that act intelligently, Searle's arguments are not usually considered an issue for AI research. Stuart Russell and Peter Norvig observe that most AI researchers "don't care about the strong AI hypothesis—as long as the program works, they don't care whether you call it a simulation of intelligence or real intelligence."

Searle's "strong AI" should not be confused with "strong AI" as defined by Ray Kurzweil and other futurists, who use the term to describe machine intelligence that rivals or exceeds human intelligence. Kurzweil is concerned primarily

with the amount of intelligence displayed by the machine, whereas Searle's argument sets no limit on this, as long as it is understood that it is a simulation and not the real thing.

System Reply

These replies attempt to answer the question: since the man in the room doesn't speak Chinese, where is the "mind" that does? These replies address the key ontological issues of mind vs. body and simulation vs. reality. All of the replies that identify the mind in the room are versions of "the system reply".

The basic "system reply" argues that it is the "whole system" which understands Chinese. While the man understands only English, when he is combined with the program, scratch paper, pencils and file cabinets, they form a system that can understand Chinese. "Here, understanding is not being ascribed to the mere individual; rather it is being ascribed to this whole system of which he is a part" Searle explains. The fact that man does not understand Chinese is irrelevant, because it is only the system as a whole which matters.

Searle notes that (in this simple version of the systems reply) there is nothing more than a list of physical objects; it grants the power of understanding and consciousness to "the conjunction of that person and bits of paper". Searle responds by simplifying the list of physical objects: he asks what happens if the man memorizes the rules and keeps track of everything in his head? Then the whole system consists of just one object: the man himself. Searle argues that if the man doesn't understand Chinese then the system doesn't understand Chinese either and the fact that the man appears to understand Chinese proves nothing. Critics of Searle's response argue that the program has allowed the man to have two minds in one head.[who?]

More sophisticated versions of the system reply try to identify more precisely what "the system" is and they differ in exactly how they describe it. According to these replies,[who?] the "mind that speaks Chinese" could be such things as: the "software", a "program", a "running program", a simulation of the "neural correlates of consciousness", the "functional system", a "simulated mind", an "emergent property", or "a virtual mind" (Marvin Minsky's version of the system reply, described below).

Virtual Mind Reply

The term "virtual" is used in computer science to describe an object which appears to exist "in" a computer (or computer network) only because software is making it appear to exist. The objects "inside" computers (including files, folders, and so on) are all "virtual", except for the computer's electronic components. Similarly, Minsky argues, a computer may contain a "mind" that is virtual in the same sense as virtual machines, virtual communities and virtual reality.

To clarify the distinction between the simple systems reply given above and virtual mind reply, David Cole notes that two simulations could be running on

one system at the same time; one speaking Chinese and one speaking Korean. While there is only one system, there can be multiple "virtual minds."

Searle responds that such a mind is, at best, a simulation, and writes: "No one supposes that computer simulations of a five-alarm fire will burn the neighborhood down or that a computer simulation of a rainstorm will leave us all drenched."

Nicholas Fearn responds that, for some things, simulation is as good as the real thing. "When we call up the pocket calculator function on a desktop computer, the image of a pocket calculator appears on the screen. We don't complain that 'it isn't really a calculator', because the physical attributes of the device do not matter." The question is, is the human mind like the pocket calculator, essentially composed of information? Or is the mind like the rainstorm, something other than a computer, and not realizable in full by a computer simulation? (The issue of simulation is also discussed in the article synthetic intelligence.)

These replies provide an explanation of exactly who it is that understands Chinese. If there is something besides the man in the room that can understand Chinese, Searle can't argue that (1) the man doesn't understand Chinese, therefore (2) nothing in the room understands Chinese. This, according to those who make this reply, shows that Searle's argument fails to prove that "strong AI" is false.

However, the replies, by themselves, do not prove that strong AI is true, either: they provide no evidence that the system (or the virtual mind) understands Chinese, other than the hypothetical premise that it passes the Turing Test. As Searle writes "the systems reply simply begs the question by insisting that system must understand Chinese."

Finding the Meaning

As far as the person in the room is concerned, the symbols are just meaningless "squiggles." But if the Chinese room really "understands" what it's saying, then the symbols must get their meaning from somewhere. These arguments attempt to connect the symbols to the things they symbolize. These replies address Searle's concerns about intentionality, symbol grounding and syntax vs. semantics.

Robot Reply

Suppose that instead of a room, the program was placed into a robot that could wander around and interact with its environment. This would allow a "causal connection" between the symbols and things they represent. Hans Moravec comments: 'If we could graft a robot to a reasoning program, we wouldn't need a person to provide the meaning anymore: it would come from the physical world."

Searle’s reply is to suppose that, unbeknownst to the individual in the Chinese room, some of the inputs came directly from a camera mounted on a robot, and

some of the outputs were used to manipulate the arms and legs of the robot. Nevertheless, the person in the room is still just following the rules, and does not know what the symbols mean. Searle writes "he doesn't see what comes into the robot's eyes."(See Mary's room for a similar thought experiment.)

Derived Meaning

Some respond that the room, as Searle describes it, is connected to the world: through the Chinese speakers that it is "talking" to and through the programmers who designed the knowledge base in his file cabinet. The symbols Searle manipulates are already meaningful, they're just not meaningful to him.

Searle says that the symbols only have a "derived" meaning, like the meaning of words in books. The meaning of the symbols depends on the conscious understanding of the Chinese speakers and the programmers outside the room. The room, according to Searle, has no understanding of its own.

Commonsense Knowledge

Some have argued that the meanings of the symbols would come from a vast "background" of commonsense knowledge encoded in the program and the filing cabinets. This would provide a "context" that would give the symbols their meaning.

Searle agrees that this background exists, but he does not agree that it can be built into programs. Hubert Dreyfus has also criticized the idea that the "background" can be represented symbolically.

To each of these suggestions, Searle's response is the same: no matter how much knowledge is written into the program and no matter how the program is connected to the world, he is still in the room manipulating symbols according to rules. His actions are syntactic and this can never explain to him what the symbols stand for. Searle writes "syntax is insufficient for semantics."

However, for those who accept that Searle's actions simulate a mind, separate from his own, the important question is not what the symbols mean to Searle, what is important is what they mean to the virtual mind. While Searle is trapped in the room, the virtual mind is not: it is connected to the outside world through the Chinese speakers it speaks to, through the programmers who gave it world knowledge, and through the cameras and other sensors that roboticists can supply.

Brain Simulator Reply

Suppose that the program simulated in fine detail the action of every neuron in the brain of a Chinese speaker. This strengthens the intuition that there would be no significant difference between the operation of the program and the operation of a live human brain.

Searle replies that such a simulation will not have reproduced the important features of the brain — its causal and intentional states. Searle is adamant that

"human mental phenomena [are] dependent on actual physical-chemical properties of actual human brains." Moreover, he argues:

"[I]magine that instead of a monolingual man in a room shuffling symbols we have the man operate an elaborate set of water pipes with valves connecting them. When the man receives the Chinese symbols, he looks up in the program, written in English, which valves he has to turn on and off. Each water connection corresponds to a synapse in the Chinese brain, and the whole system is rigged up so that after doing all the right firings, that is after turning on all the right faucets, the Chinese answers pop out at the output end of the series of pipes. Now where is the understanding in this system? It takes Chinese as input, it simulates the formal structure of the synapses of the Chinese brain, and it gives Chinese as output. But the man certainly doesn't understand Chinese, and neither do the water pipes, and if we are tempted to adopt what I think is the absurd view that somehow the conjunction of man and water pipes understands, remember that in principle the man can internalize the formal structure of the water pipes and do all the "neuron firings" in his imagination. " Searle (1980)

Formal Arguments

Searle has produced a more formal version of the argument of which the Chinese Room forms a part. He presented the first version in 1984. The version given below is from 1990. The part of the argument which should be controversial is A3 and it is this point which the Chinese room thought experiment is intended to prove.

He begins with three axioms:

(A1) "Programs are formal (syntactic)."

A program uses syntax to manipulate symbols and pays no attention to the semantics of the symbols. It knows where to put the symbols and how to move them around, but it doesn't know what they stand for or what they mean. For the program, the symbols are just physical objects like any others.

(A2) "Minds have mental contents (semantics)."

Unlike the symbols used by a program, our thoughts have meaning: they represent things and we know what it is they represent.

(A3) "Syntax by itself is neither constitutive of nor sufficient for semantics."

This is what the Chinese room argument is intended to prove: the Chinese room has syntax (because there is a man in there moving symbols around). The Chinese room has no semantics (because, according to Searle, there is no one or nothing in the room that understands what the symbols mean). Therefore, having syntax is not enough to generate semantics.

Searle posits that these lead directly to this conclusion:

(C1) Programs are neither constitutive of nor sufficient for minds.

This should follow without controversy from the first three: Programs don't have semantics. Programs have only syntax, and syntax is insufficient for semantics. Every mind has semantics. Therefore programs are not minds.

This much of the argument is intended to show that artificial intelligence will never produce a machine with a mind by writing programs that manipulate symbols. The remainder of the argument addresses a different issue. Is the human brain running a program? In other words, is the computational theory of mind correct? He begins with an axiom that is intended to express the basic modern scientific consensus about brains and minds:

(A4) Brains cause minds.

Searle claims that we can derive "immediately" and "trivially" that:

(C2) Any other system capable of causing minds would have to have causal powers (at least) equivalent to those of brains.

Brains must have something that causes a mind to exist. Science has yet to determine exactly what it is, but it must exist, because minds exist. Searle calls it "causal powers". "Causal powers" is whatever the brain uses to create a mind. If anything else can cause a mind to exist, it must have "equivalent causal powers". "Equivalent causal powers" is whatever else that could be used to make a mind.

And from this he derives the further conclusions:

(C3) Any artifact that produced mental phenomena, any artificial brain, would have to be able to duplicate the specific causal powers of brains, and it could not do that just by running a formal program.

This follows from C1 and C2: Since no program can produce a mind, and "equivalent causal powers" produce minds, it follows that programs do not have "equivalent causal powers."

(C4) The way that human brains actually produce mental phenomena cannot be solely by virtue of running a computer program.

Since programs do not have "equivalent causal powers", "equivalent causal powers" produce minds, and brains produce minds, it follows that brains do not use programs to produce minds.

Mind Uploading

Whole brain emulation or mind uploading (sometimes called mind transfer) is the hypothetical process of transferring or copying a conscious mind from a brain to a non-biological substrate by scanning and mapping a biological brain in detail and copying its state into a computer system or another computational device. The computer would have to run a simulation model so faithful to the original that it would behave in essentially the same way as the original brain, or for all practical purposes, indistinguishably. The simulated mind is assumed to be part of a virtual reality simulated world, supported by an anatomic 3D body simulation model. Alternatively, the simulated mind could be assumed to reside in a computer inside (or connected to) a humanoid robot or a biological body, replacing its brain.

Whole brain emulation is discussed by futurists as a "logical endpoint" of the topical computational neuroscience and neuroinformatics fields, both about brain simulation for medical research purposes. It is discussed in artificial intelligence research publications as an approach to strong AI. Among futurists and within the transhumanist movement it is an important proposed life extension technology, originally suggested in biomedical literature in 1971. It is a central conceptual feature of numerous science fiction novels and films.

Whole brain emulation is considered by some scientists as a theoretical and futuristic but possible technology, although mainstream research funders and scientific journals remain skeptical. Several contradictory predictions have been made about when a whole human brain can be emulated; some of the predicted dates have already passed. Substantial mainstream research and development are however being done in relevant areas including development of faster super computers, virtual reality, brain-computer interfaces, animal brain mapping and simulation, connectomics and information extraction from dynamically functioning brains.

The question whether an emulated brain can be a human mind is debated by philosophers, and may be viewed as impossible by those who hold a dualistic view of the human mind, which is common in many religions.

Simple Artificial Neural Network

The human brain contains about 100 billion nerve cells called neurons, each individually linked to other neurons by way of connectors called axons and dendrites. Signals at the junctures (synapses) of these connections are transmitted by the release and detection of chemicals known as neurotransmitters. The established neuroscientific consensus is that the human mind is largely an emergent property of the information processing of this neural network.

Importantly, neuroscientists have stated that important functions performed by the mind, such as learning, memory, and consciousness, are due to purely physical and electrochemical processes in the brain and are governed by

applicable laws. For example, Christof Koch and Giulio Tononi wrote in IEEE Spectrum:

"Consciousness is part of the natural world. It depends, we believe, only on mathematics and logic and on the imperfectly known laws of physics, chemistry, and biology; it does not arise from some magical or otherworldly quality."

The concept of mind uploading is based on this mechanistic view of the mind, and denies the vitalist view of human life and consciousness.

Many eminent computer scientists and neuroscientists have predicted that computers will be capable of thought and even attain consciousness, including Koch and Tononi, Douglas Hofstadter, Jeff Hawkins, Marvin Minsky, Randal A. Koene, and Rodolfo Llinas.

Such a machine intelligence capability might provide a computational substrate necessary for uploading.

However, even though uploading is dependent upon such a general capability it is conceptually distinct from general forms of AI in that it results from dynamic reanimation of information derived from a specific human mind so that the mind retains a sense of historical identity (other forms are possible but would compromise or eliminate the life-extension feature generally associated with uploading). The transferred and reanimated information would become a form of artificial intelligence, sometimes called an infomorph or "noömorph."

Many theorists have presented models of the brain and have established a range of estimates of the amount of computing power needed for partial and complete simulations. Using these models, some have estimated that uploading may become possible within decades if trends such as Moore's Law continue.

The prospect of uploading human consciousness in this manner raises many philosophical questions involving identity, individuality and if the soul and mind can be defined as the information content of the brain, as well as numerous problems of medical ethics and morality of the process.

Immortality/Backup

In theory, if the information and processes of the mind can be disassociated from the biological body, they are no longer tied to the individual limits and lifespan of that body. Furthermore, information within a brain could be partly or wholly copied or transferred to one or more other substrates (including digital storage or another brain), thereby reducing or eliminating mortality risk. This general proposal appears to have been first made in the biomedical literature in 1971 by biogerontologist George M. Martin of the University of Washington.

Speedup

A computer-based intelligence such as an upload could potentially think much faster than a human even if it were no more intelligent. Human neurons

exchange electrochemical signals with a maximum speed of about 150 meters per second, whereas the speed of light is about 300 million meters per second, about two million times faster. Also, neurons can generate a maximum of about 200 to 1000 action potentials or "spikes" per second, whereas the number of signals per second in modern[when?] computer chips is about 3 GHz (about 20 million times greater) and expected to increase by at least a factor 100. Therefore, even if the computer components responsible for simulating a brain were not significantly smaller than a biological brain, and even if the temperature of these components was not significantly lower, Eliezer Yudkowsky of the Singularity Institute for Artificial Intelligence calculates a theoretical upper bound for the speed of a future artificial neural network. It could in theory run about 1 million times faster than a real brain, experiencing about a year of subjective time in only 31 seconds of real time.

However, in practice this massively parallel implementation would require separate computational units for each of the hundred billion neurons and each of the hundred trillion synapses. That requires an enormously large computer or artificial neural network in comparison with today's super-computers. In a less futuristic implementation, time-sharing would allow several neurons to be emulated sequentially by the same computational unit. Thus the size of the computer would be restricted, but the speedup would be lower. Assuming that cortical minicolumns organized into hypercolumns are the computational units, mammal brains can be emulated by today's super computers, but with slower speed than in a biological brain.

Multiple / Parallel Existence

Another concept explored in science fiction is the idea of more than one running "copy" of a human mind existing at once. Such copies could potentially allow an "individual" to experience many things at once, and later integrate the experiences of all copies into a central mentality at some point in the future, effectively allowing a single sentient being to "be many places at once" and "do many things at once"; this concept has been explored in fiction. Such partial and complete copies of a sentient being raise interesting questions regarding identity and individuality.

Computational Capacity

Futurist Ray Kurzweil's projected supercomputer processing power based on Moore's law exponential development of computing capacity. Here the computational capacity doubling time is assumed to be 1.2 years.

Advocates of mind uploading point to Moore's law to support the notion that the necessary computing power is expected to become available within a few decades. However, the actual computational requirements for running an

uploaded human mind are very difficult to quantify, potentially rendering such an argument specious.

Regardless of the techniques used to capture or recreate the function of a human mind, the processing demands are likely to be immense, due to the large number of neurons in the human brain along with the considerable complexity of each neuron.

In 2004, Henry Markram, lead researcher of the "Blue Brain Project", has stated that "it is not [their] goal to build an intelligent neural network", based solely on the computational demands such a project would have.

It will be very difficult because, in the brain, every molecule is a powerful computer and we would need to simulate the structure and function of trillions upon trillions of molecules as well as all the rules that govern how they interact. You would literally need computers that are trillions of times bigger and faster than anything existing today.

Five years later, after successful simulation of part of a rat brain, the same scientist was much more bold and optimistic. In 2009, when he was director of the Blue Brain Project, he claimed that a detailed, functional artificial human brain can be built within the next 10 years

Simulation Model Scale

A high-level cognitive AI model of the brain architecture is not required for brain emulation.

Simple neuron model: Black-box dynamic non-linear signal processing system

Metabolism model: The movement of positively-charged ions through the ion channels controls the membrane electrical action potential in an axon.

Since the function of the human mind, and how it might arise from the working of the brain's neural network, are poorly understood issues, mind uploading relies on the idea of neural network emulation. Rather than having to understand the high-level psychological processes and large-scale structures of the brain, and model them using classical artificial intelligence methods and cognitive psychology models, the low-level structure of the underlying neural network is captured, mapped and emulated with a computer system. In computer science terminology, rather than analyzing and reverse engineering the behavior of the algorithms and data structures that resides in the brain, a blueprint of its source code is translated to another programming language. The human mind and the personal identity then, theoretically, is generated by the emulated neural network in an identical fashion to it being generated by the biological neural network.

On the other hand, a molecule-scale simulation of the brain is not expected to be required, provided that the functioning of the neurons is not affected by quantum mechanical processes. The neural network emulation approach only requires that the functioning and interaction of neurons and synapses are

understood. It is expected that it is sufficient with a black-box signal processing model of how the neurons respond to nerve impulses (electrical as well as chemical synaptic transmission).

A sufficiently complex and accurate model of the neurons is required. A traditional artificial neural network model, for example multi-layer perceptron network model, is not considered as sufficient. A dynamic spiking neural network model is required, which reflects that the neuron fires only when a membrane potential reaches a certain level. It is likely that the model must include delays, non-linear functions and differential equations describing the relation between electrophysical parameters such as electrical currents, voltages, membrane states (ion channel states) and neuromodulators.

Since learning and long-term memory are believed to result from strengthening or weakening the synapses via a mechanism known as synaptic plasticity or synaptic adaptation, the model should include this mechanism. The response of sensory receptors to various stimuli must also be modeled.

Furthermore, the model may have to include metabolism, i.e. how the neurons are affected by hormones and other chemical substances that may cross the blood-brain barrier. It is considered likely that the model must include currently unknown neuromodulators, neurotransmitters and ion channels. It is considered unlikely that the simulation model has to include protein interaction, which would make it computationally complex.

A digital computer simulation model of an analog system such as the brain is an approximation that introduces random quantization errors and distortion. However, the biological neurons also suffer from randomness and limited precision, for example due to background noise. The errors of the discrete model can be made smaller than the randomness of the biological brain by choosing a sufficiently high variable resolution and sample rate, and sufficiently accurate models of non-linearities. The computational power and computer memory must however be sufficient to run such large simulations, preferably in real time.

Scanning and Mapping Scale of an Individual

When modelling and simulating the brain of a specific individual, a brain map or connectivity database showing the connections between the neurons must be extracted from an anatomic model of the brain. This network map should show the connectivity of the whole nervous system, including the spinal cord, sensory receptors, and muscle cells.

Destructive scanning of the human brain including synaptic details is possible as of end of 2010. A full brain map should also reflect the synaptic strength (the "weight") of each connection. It is unclear if the current technology allows that.

It is proposed that short-term memory and working memory is prolonged or repeated firing of neurons, as well as intra-neural dynamic processes. Since the

electrical and chemical signal state of the synapses and neurons may be hard to extract, the uploading might result in that the uploaded mind perceives a memory loss of the events immediately before the time of brain scanning.

A full brain map would occupy less than 2 x 1016 bytes (20000 TB) and would store the addresses of the connected neurons, the synapse type and the synapse "weight" for each of the brains' 1015 synapses.

Serial Sectioning of a Brain

A possible method for mind uploading is serial sectioning, in which the brain tissue and perhaps other parts of the nervous system are frozen and then scanned and analyzed layer by layer, thus capturing the structure of the neurons and their interconnections. The exposed surface of frozen nerve tissue would be scanned and recorded, and then the surface layer of tissue removed. While this would be a very slow and labor intensive process, research is currently underway to automate the collection and microscopy of serial sections. The scans would then be analyzed, and a model of the neural net recreated in the system that the mind was being uploaded into.

There are uncertainties with this approach using current microscopy techniques. If it is possible to replicate neuron function from its visible structure alone, then the resolution afforded by a scanning electron microscope would suffice for such a technique. However, as the function of brain tissue is partially determined by molecular events (particularly at synapses, but also at other places on the neuron's cell membrane), this may not suffice for capturing and simulating neuron functions. It may be possible to extend the techniques of serial sectioning and to capture the internal molecular makeup of neurons, through the use of sophisticated immunohistochemistry staining methods which could then be read via confocal laser scanning microscopy. However, as the physiological genesis of 'mind' is not currently known, this method may not be able to access all of the necessary biochemical information to recreate a human brain with sufficient fidelity.

Brain Imaging

It may also be possible to create functional 3D maps of the brain activity, using advanced neuroimaging technology, such as functional MRI (fMRI, for mapping change in blood flow), Magnetoencephalography (MEG, for mapping of electrical currents), or combinations of multiple methods, to build a detailed three-dimensional model of the brain using non-invasive and non-destructive methods. Today, fMRI is often combined with MEG for creating functional maps of human cortex during more complex cognitive tasks, as the methods complement each other. Even though current imaging technology lacks the spatial resolution needed to gather the information needed for such a scan, important recent and future developments are predicted to substantially improve both spatial and temporal resolutions of existing technologies.

Brain-Computer Interface (BCI)

Brain-computer interfaces (BCI) (also known as neuro-computer interfaces, direct neuron interfaces or cerebral interfaces) constitute one of the hypothetical technologies for the reading of information in the dynamically functioning brain. The production of this or a similar device may be essential to the possibility of mind uploading a living human subject.

Neuroinformatics

An artificial neural network described as being "as big and as complex as half of a mouse brain" was run on an IBM blue gene supercomputer by a University of Nevada research team in 2007. A simulated time of one second took ten seconds of computer time. The researchers said they had seen "biologically consistent" nerve impulses flowed through the virtual cortex. However, the simulation lacked the structures seen in real mice brains, and they intend to improve the accuracy of the neuron model.

Blue Brain is a project, launched in May 2005 by IBM and the Swiss Federal Institute of Technology in Lausanne, with the aim to create a computer simulation of a mammalian cortical column, down to the molecular level. The project uses a supercomputer based on IBM's Blue Gene design to simulate the electrical behavior of neurons based upon their synaptic connectivity and complement of intrinsic membrane currents. The initial goal of the project, completed in December 2006, was the simulation of a rat neocortical column, which can be considered the smallest functional unit of the neocortex (the part of the brain thought to be responsible for higher functions such as conscious thought), containing 10,000 neurons (and 108 synapses). Between 1995 and 2005, Henry Markram mapped the types of neurons and their connections in such a column. In November 2007, the project reported the end of the first phase, delivering a data-driven process for creating, validating, and researching the neocortical column. The project seeks to eventually reveal aspects of human cognition and various psychiatric disorders caused by malfunctioning neurons, such as autism, and to understand how pharmacological agents affect network behavior.

An organization called the Brain Preservation Foundation was founded in 2010 and is offering a Brain Preservation Technology prize to promote exploration of brain preservation technology in service of humanity. The Prize, currently $106,000, will be awarded in two parts, 25% to the first international team to preserve a whole mouse brain, and 75% to the first team to preserve a whole large animal brain in a manner that could also be adopted for humans in a hospital or hospice setting immediately upon clinical death. Ultimately the goal of this prize is to generate a whole brain map which may be used in support of separate efforts to upload and possibly 'reboot' a mind in virtual space.

Legal, Political and Economical Issues

It may be difficult to ensure the protection of human rights in simulated worlds. For example, social science researchers might be tempted to expose simulated minds, or whole isolated societies of simulated minds, to controlled

experiments in which many copies of the same minds are exposed (serially or simultaneously) to different test conditions.

The only limited physical resource to be expected in a simulated world is the computational capacity, and thus the speed and complexity of the simulation. Wealthy or privileged individuals in a society of uploads might thus experience more subjective time than others in the same real time, or may be able to run multiple copies of themselves or others, and thus produce more service and become even more wealthy. Others may suffer from computational resource starvation and show a slow motion behavior.

Copying vs. Moving

Another philosophical issue with mind uploading is whether an uploaded mind is really the "same" sentience, or simply an exact copy with the same memories and personality; or, indeed, what the difference could be between such a copy and the original (see the Swampman thought experiment). This issue is especially complex if the original remains essentially unchanged by the procedure, thereby resulting in an obvious copy which could potentially have rights separate from the unaltered, obvious original.

Most projected brain scanning technologies, such as serial sectioning of the brain, would necessarily be destructive, and the original brain would not survive the brain scanning procedure. But if it can be kept intact, the computer-based consciousness could be a copy of the still-living biological person. It is in that case implicit that copying a consciousness could be as feasible as literally moving it into one or several copies, since these technologies generally involve simulation of a human brain in a computer of some sort, and digital files such as computer programs can be copied precisely. It is usually assumed that once the versions are exposed to different sensory inputs, their experiences would begin to diverge, but all their memories up until the moment of the copying would remain the same.

The problem is made even more serious by the possibility of creating a potentially infinite number of initially identical copies of the original person, which would of course all exist simultaneously as distinct beings. The most parsimonious view of this phenomenon is that the two (or more) minds would share memories of their past but from the point of duplication would simply be distinct minds (although this is complicated by merging). Many complex variations are possible.

Depending on computational capacity, the simulation may run at faster or slower simulation time as compared to the elapsed physical time, resulting in that the simulated mind would perceive that the physical world is running in slow motion or fast motion respectively, while biological persons will see the simulated mind in fast or slow motion respectively.

A brain simulation can be started, paused, backed-up and rerun from a saved backup state at any time. The simulated mind would in the latter case forget everything that has happened after the instant of backup, and perhaps not even

be aware that it is repeating itself. An older version of a simulated mind may meet a younger version and share experiences with it.

Bekenstein Bound

The Bekenstein bound is an upper limit on information that can be contained within a given finite region of space which has a finite amount of energy or, conversely, the maximum amount of information required to perfectly describe a given physical system down to the quantum level.

An average human brain has a weight of 1.5 kg and a volume of 1260 cm3. The energy ($E = m \cdot c2$) will be $1.34813 \cdot 1017$ J and if the brain is approximate to a sphere then the radius ($V = 4 \cdot \pi \cdot r3/3$) will be $6.70030 \cdot 10{-}2$ m.

The Bekenstein bound ($I \leq$

$$\frac{2 \cdot \pi \cdot r \cdot E}{\hbar \cdot c \cdot \ln 2}$$

) for an average human brain would be $2.58991 \cdot 1042$ bit and represents an upper bound on the information needed to perfectly recreate the average human brain down to the quantum level. This implies that the number of different states ($\Omega=2I$) of the human brain (and of the mind if the physicalism is true) is at most $107.79640 \cdot 1041$.

However, as described above, many mind uploading advocates expect that quantum-level models and molecule-scale simulation of the neurons will not be needed, so the Bekenstein bound only represents a maximum upper limit. The Hippocampus of a human adult brain has been estimated to store a limit of up to 2.5 petabyte of binary data equivalent.

Followers of the Raëlian religion advocate mind uploading in the process of human cloning to achieve eternal life. Living inside of a computer is also seen by followers as an eminent possibility.

However, mind uploading is also advocated by a number of secular researchers in neuroscience and artificial intelligence, such as Marvin Minsky. In 1993, Joe Strout created a small web site called the Mind Uploading Home Page, and began advocating the idea in cryonics circles and elsewhere on the net. That site has not been actively updated in recent years, but it has spawned other sites including MindUploading.org, run by Randal A. Koene, Ph.D., who also moderates a mailing list on the topic. These advocates see mind uploading as a medical procedure which could eventually save countless lives.

Many transhumanists look forward to the development and deployment of mind uploading technology, with transhumanists such as Nick Bostrom predicting that it will become possible within the 21st century due to technological trends such as Moore's Law.

The book Beyond Humanity: CyberEvolution and Future Minds by Gregory S. Paul & Earl D. Cox, is about the eventual (and, to the authors, almost

inevitable) evolution of computers into sentient beings, but also deals with human mind transfer. Richard Doyle's Wetwares: Experiments in PostVital Living deals extensively with uploading from the perspective of distributed embodiment, arguing for example that humans are currently part of the "artificial life phenotype." Doyle's vision reverses the polarity on uploading, with artificial life forms such as uploads actively seeking out biological embodiment as part of their reproductive strategy. Raymond Kurzweil, a prominent advocate of transhumanism and the likelihood of a technological singularity, has suggested that the easiest path to human-level artificial intelligence may lie in "reverse-engineering the human brain", which he usually uses to refer to the creation of a new intelligence based on the general "principles of operation" of the brain, but he also sometimes uses the term to refer to the notion of uploading individual human minds based on highly detailed scans and simulations. This idea is discussed on pp. 198–203 of his book The Singularity is Near, for example.

Bots

Internet bots, also known as web robots, WWW robots or simply bots, are software applications that run automated tasks over the Internet. Typically, bots perform tasks that are both simple and structurally repetitive, at a much higher rate than would be possible for a human alone. The largest use of bots is in web spidering, in which an automated script fetches, analyzes and files information from web servers at many times the speed of a human. Each server can have a file called robots.txt, containing rules for the spidering of that server that the bot is supposed to obey.

Chatter Bot

A chatter robot, chatterbot, chatbot, or chat bot is a computer program designed to simulate an intelligent conversation with one or more human users via auditory or textual methods, primarily for engaging in small talk. The primary aim of such simulation has been to fool the user into thinking that the program's output has been produced by a human (the Turing test). Programs playing this role are sometimes referred to as Artificial Conversational Entities, talk bots or chatterboxes. In addition, however, chatterbots are often integrated into dialog systems for various practical purposes such as online help, personalised service, or information acquisition. Some chatterbots use sophisticated natural language processing systems, but many simply scan for keywords within the input and pull a reply with the most matching keywords, or the most similar wording pattern, from a textual database.

The term "ChatterBot" was originally coined by Michael Mauldin (Creator of the first Verbot, Julia) in 1994 to describe these conversational programs.

In 1950, Alan Turing published his famous article "Computing Machinery and Intelligence", which proposed what is now called the Turing test as a criterion of intelligence. This criterion depends on the ability of a computer program to

impersonate a human in a real-time written conversation with a human judge, sufficiently well that the judge is unable to distinguish reliably—on the basis of the conversational content alone—between the program and a real human.

ELIZA

The notoriety of Turing's proposed test stimulated great interest in Joseph Weizenbaum's program ELIZA, published in 1966, which seemed to be able to fool users into believing that they were conversing with a real human. However Weizenbaum himself did not claim that ELIZA was genuinely intelligent, and the Introduction to his paper presented it more as a debunking exercise:

[In] artificial intelligence ... machines are made to behave in wondrous ways, often sufficient to dazzle even the most experienced observer. But once a particular program is unmasked, once its inner workings are explained ... its magic crumbles away; it stands revealed as a mere collection of procedures ... The observer says to himself "I could have written that". With that thought he moves the program in question from the shelf marked "intelligent", to that reserved for curios ... The object of this paper is to cause just such a re-evaluation of the program about to be "explained". Few programs ever needed it more.

ELIZA's key method of operation (copied by chatbot designers ever since) involves the recognition of cue words or phrases in the input, and the output of corresponding pre-prepared or pre-programmed responses that can move the conversation forward in an apparently meaningful way (e.g. by responding to any input that contains the word 'MOTHER' with 'TELL ME MORE ABOUT YOUR FAMILY'). Thus an illusion of understanding is generated, even though the processing involved has been merely superficial. ELIZA showed that such an illusion is surprisingly easy to generate, because human judges are so ready to give the benefit of the doubt when conversational responses are capable of being interpreted as "intelligent". Thus the key technique here—which characterises a program as a chatbot rather than as a serious natural language processing system—is the production of responses that are sufficiently vague and non-specific that they can be understood as "intelligent" in a wide range of conversational contexts. The emphasis is typically on vagueness and unclarity, rather than any conveying of genuine information.

Interface designers have come to appreciate that humans' readiness to interpret computer output as genuinely conversational—even when it is actually based on rather simple pattern-matching—can be exploited for useful purposes. Most people prefer to engage with programs that are human-like, and this gives chatbot-style techniques a potentially useful role in interactive systems that need to elicit information from users, as long as that information is relatively straightforward and falls into predictable categories. Thus, for example, online help systems can usefully employ chatbot techniques to identify the area of help that users require, potentially providing a "friendlier" interface than a

more formal search or menu system. This sort of usage holds the prospect of moving chatbot technology from Weizenbaum's "shelf ... reserved for curios" to that marked "genuinely useful computational methods".

ActiveBuddy was a company that created conversation-based interactive agents originally distributed via instant messaging platforms. Founded in 2000, the company was the brainchild of Robert Hoffer and Timothy Kay. The idea for interactive agents came from Hoffer's vision to add functionality to increasingly popular instant messaging services. The original implementation took shape as a word-based adventure game but quickly grew to include a wide range of database applications including access to news, weather, stock information, movie times, yellow pages listings, and detailed sports data, as well as a variety of tools (calculators, translator, etc.). These various applications were bundled into a single entity and launched as SmarterChild in 2001. SmarterChild acted as a showcase for the quick data access and possibilities for fun conversation that the company planned to turn into customized, niche specific products.

The rapid success of SmarterChild, over 13 million users, led to targeted promotional products for Radiohead, Austin Powers, The Sporting News, and others. ActiveBuddy sought to strengthen its hold on the interactive agent market for the future by filing for, and receiving, a patent on their creation in 2002.The company also released the BuddyScript SDK, a free developer kit that allow programmers to design and launch their own interactive agents using ActiveBuddy's proprietary scripting language, in 2002. The company subsequently changed its name from ActiveBuddy to Conversagent in 2003, and then again to Colloquis in 2006. Colloquis was later purchased by Microsoft in October of 2006.

Development

The classic historic early chatterbots are ELIZA (1966) and PARRY (1972). More recent notable programs include A.L.I.C.E., Jabberwacky and D.U.D.E (Agence Nationale de la Recherche and CNRS 2006). While ELIZA and PARRY were used exclusively to simulate typed conversation, many chatterbots now include functional features such as games and web searching abilities. In 1984, a book called The Policeman's Beard is Half Constructed was published, allegedly written by the chatbot Racter (though the program as released would not have been capable of doing so).

One pertinent field of AI research is natural language processing. Usually, weak AI fields employ specialized software or programming languages created specifically for the narrow function required. For example, A.L.I.C.E. utilises a programming language called AIML, which is specific to its function as a conversational agent, and has since been adopted by various other developers of, so called, Alicebots. Nevertheless, A.L.I.C.E. is still purely based on pattern matching techniques without any reasoning capabilities, the same technique

ELIZA was using back in 1966. This is not strong AI, which would require sapience and logical reasoning abilities.

Jabberwacky learns new responses and context based on real-time user interactions, rather than being driven from a static database. Some more recent chatterbots also combine real-time learning with evolutionary algorithms that optimise their ability to communicate based on each conversation held, with one notable example being Kyle, winner of the 2009 Leodis AI Award. Still, there is currently no general purpose conversational artificial intelligence, and some software developers focus on the practical aspect, information retrieval.

Chatterbot competitions focus on the Turing test or more specific goals. Two such annual contests are the Loebner Prize and The Chatterbox Challenge.

IRC Bot

An IRC bot is a set of scripts or an independent program that connects to Internet Relay Chat as a client, and so appears to other IRC users as another user. An IRC bot differs from a regular client in that instead of providing interactive access to IRC for a human user, it performs automated functions.

The historically oldest IRC bots were Bill Wisner's Bartender and Greg Lindahl's GM (Game Manager for the Hunt the Wumpus game). Over time, bots evolved to provide special services, such as managing channels on behalf of groups of users, maintaining access lists, and providing access to databases.

Often, an IRC bot is deployed as a detached program running from a stable host. It sits on an IRC channel to keep it open and prevents malicious users from taking over the channel. It can be configured to give channel operator status to privileged users when they join the channel, and can provide a unified channel operator list. Many of these features require that the bot be a channel operator. Thus, most IRC bots are run from computers which have long uptimes (generally running a BSD derivative or Linux) and a fast, stable Internet connection. As IRC has become popular with many dial-up users as well, special services have appeared that offer limited user-level access to a stable Linux server with a decent connection. The user may run an IRC bot from this shell account. These services are commonly known as shell providers.

A bot can also perform many other useful functions, such as logging what happens in an IRC channel, giving out information on demand (very popular in IRC channels dealing with user support), creating statistics, hosting trivia games, and so on. These functions are usually provided by user-writable scripts, often written in a scripting programming language such as Tcl or Perl, added to the bot in question. Channels dedicated to file sharing often use XDCC bots to distribute their files.

IRC bots are particularly well-used on IRC networks without channel registration services like ChanServ, such as EFnet and IRCnet, and on networks that may prevent channels from being registered due to certain

registration requirements (minimum user count, etc.), such as Undernet or QuakeNet. Where bots are used for administrative functions such as this, they may need more access than a normal client connection allows. Some versions of IRC have a "Service" protocol that allows clients with these extra powers. Such server-sanctioned bots are called IRC services.

Bots are not always welcome. Some IRC networks forbid the usage of bots. One of the reasons for doing so is that each nickname connected to the network increases the size of the network database which is being kept in sync across all servers. Allowing for bots in large networks can cause a relevant amount of network traffic overhead which needs to be financed and may even lead to netsplits. This however is a shortcoming of the IRC technology, not the bots.

People that create an IRC bot use either the scripting language built into a client, or appropriate frameworks of a suitable programming language to connect to an IRC server(s), or they use an existing bot implementation, and adapt it to their needs.

In computing, an avatar is the graphical representation of the user or the user's alter ego or character. It may take either a three-dimensional form, as in games or virtual worlds, or a two-dimensional form as an icon in Internet forums and other online communities. It can also refer to a text construct found on early systems such as MUDs. It is an object representing the user. The term "avatar" can also refer to the personality connected with the screen name, or handle, of an Internet user.

Avatar

The use of the term avatar for the on-screen representation of the user was coined in 1985 by Chip Morningstar and Joseph Romero in designing LucasFilm's online role-playing game Habitat. The computer game Ultima IV: Quest of the Avatar was released in 1985, but does not use the word in this sense, only in its original religious sense; the player's in-game final objective is to become an "Avatar". Later games in the Ultima series use the term in the same sense as Habitat and introduce Habitat-style customization of avatars. Another early use of the term was in the pen and paper role-playing game Shadowrun (1989).

Norman Spinrad

In Norman Spinrad's novel Songs from the Stars (1980), the term avatar is used in a description of a computer generated virtual experience. In the story, humans receive messages from an alien galactic network that wishes to share knowledge and experience with other advanced civilizations through "songs". The humans build a "galactic receiver" that describes itself:

The galactic receiver is programmed to derive species specific full sensory input data from standard galactic meaning code equations. By controlling your sensorium input along species specific parameters galactic songs astral back-

project you into approximation of total involvement in artistically recreated broadcast realities...

From the last page of the chapter titled "The Galactic Way" in a description of an experience that is being relayed via the galactic receiver to the main characters:

You stand in a throng of multifleshed being, mind avatared in all its matter, on a broad avenue winding through a city of blue trees with bright red foliage and living buildings growing from the soil in a multitude of forms.

William Gibson

Although William Gibson's notion of cyberspace as a consensual virtual reality hallucination representing data as a 3D matrix is not remotely how users access or perceive the Internet, his cyberpunk novel Count Zero (1986) described in detail a character's representation socializing in an online world:

A square of cyberspace directly in front of him flipped sickeningly and he found himself in a pale blue graphic that seemed to represent a very spacious apartment, low shapes of furniture sketched in hair-fine lines of blue neon. A woman stood in front of him, a sort of glowing cartoon squiggle of a woman, the face a brown smudge. "I'm Slide," the figure said, hands on its hips...[She] gestured, a window suddenly snapping into existence behind her.

"Right," Bobby said. "What is this? I mean, if you could sort of explain." He still couldn't move. The "window" showed a blue-gray video view of palm trees and old buildings.

..."Hey, man, I paid a designer an arm and a leg to punch this up for me. This is my space, my construct. This is L.A., boy. People here don't do anything without jacking. This is where I entertain!"

Neal Stephenson

The use of Avatar to mean online virtual bodies was popularised by Neal Stephenson in his cyberpunk novel Snow Crash (1992). In Snow Crash, the term Avatar was used to describe the virtual simulation of the human form in the Metaverse, a fictional virtual-reality application on the Internet. Social status within the Metaverse was often based on the quality of a user's avatar, as a highly detailed avatar showed that the user was a skilled hacker and programmer while the less talented would buy off-the-shelf models in the same manner a beginner would today. Stephenson wrote in the "Acknowledgments" to Snow Crash:

The idea of a "virtual reality" such as the Metaverse is by now widespread in the computer-graphics community and is being used in a number of different ways. The particular vision of the Metaverse as expressed in this novel originated from idle discussion between me and Jaime (Captain Bandwidth) Taffe...The words avatar (in the sense used here) and Metaverse are my inventions, which I came up with when I decided that existing words (such as

virtual reality) were simply too awkward to use...after the first publication of Snow Crash, I learned that the term avatar has actually been in use for a number of years as part of a virtual reality system called Habitat...in addition to avatars, Habitat includes many of the basic features of the Metaverse as described in this book.

Artificial Intelligence

Avatars can be used as virtual embodiments of embodied agents, which are driven more or less by artificial intelligence rather than real people. Automated online assistants are examples of avatars used in this way.

Such avatars are used by organizations as a part of automated customer services in order to interact with consumers and users of services. This can avail for enterprises to reduce their operating and training cost. A major underlying technology to such systems is natural language processing. Some of these avatars are commonly known as "bots". Famous examples include Ikea's Anna, an avatar designed to guide users around the Ikea website.

Such avatars can also be powered by a Digital conversation which provides a little more structure than those using NLP, offering the user options and clearly defined paths to an outcome. This kind of avatar is known as a Structured Language Processing or SLP Avatar. Both types of avatar provide a cost effective and efficient way of engaging with consumers.

Video Games

Avatars in video games are essentially the player's physical representation in the game world. In most games, the player's representation is fixed, however increasingly games offer a basic character model, or template, and then allow customization of the physical features as the player sees fit. For example, Carl Johnson, the avatar from Grand Theft Auto: San Andreas, can be dressed in a wide range of clothing, can be given tattoos and haircuts, and can even body build or become obese depending upon player actions.

Aside from an avatar's physical appearance, its dialogue, particularly in cut scenes, may also reveal something of its character. A good example is the crude, action hero stereotype, Duke Nukem. Other avatars, such as Gordon Freeman from Half-Life, who never speaks at all, reveal very little of themselves (the original game never showed the player what he looked like without the use of a console command for third-person view).

Massively multiplayer online games (MMOGs) are the source of the most varied and sophisticated avatars. Customization levels differ between games; For example in EVE Online, players construct a wholly customized portrait, using a software that allows for several changes to facial structure as well as preset hairstyles, skin tones, etc. However, these portraits appear only in in-game chats and static information view of other players. Usually, all players appear in gigantic spacecraft that give no view of their pilot, unlike in most other RPGs. Felicia Day, creator and star of The Guild web series, created a

song called "(Do You Wanna Date My) Avatar" which satirizes avatars and virtual dating.

Game consoles such as the Wii, Playstation 3, and Xbox 360 (shown here) feature universal animated avatars.

Alternatively, City of Heroes offers one of the most detailed and comprehensive in-game avatar creation processes, allowing players to construct anything from traditional superheroes to aliens, medieval knights, monsters, robots, and many more.

Nintendo's Wii console allows for the creation of avatars called "Miis" that take the form of stylized, cartoonish people and can be used in some games as avatars for players, as in Wii Sports. In some games, the ability to use a Mii as an avatar must be unlocked, such as in Mario Kart Wii.

On November 19, 2008, Microsoft released an Xbox 360 Dashboard update which featured the introduction of Avatars as part of the console's New Xbox Experience. With the update installed users can personalize the look of their Avatars by choosing from a range of clothing and facial features. On August 11, 2009, the NXE Avatar program was updated with the inclusion of an Avatar Marketplace feature that allows users to purchase additional product and game branded clothing, jewelry, full body suits, and animated props. On initial release of the update, game branded content included items from Gears of War 2, BioShock 2, Star Wars, Fable II, Halo 3, and The Secret of Monkey Island special edition. The Xbox LIVE Avatar Marketplace is updated weekly with new items.

PlayStation Home for Sony's PlayStation 3 console also features the use of avatars, but with a more realistic style than Nintendo's Miis or Microsoft's Avatars.

Avatars in non-gaming online worlds are used as two- or three-dimensional human or fantastic representations of a person's inworld self. Such representations are a tool which facilitates the exploration of the virtual universe, or acts as a focal point in conversations with other users, and can be customized by the user. Usually, the purpose and appeal of such universes is to provide a large enhancement to common online conversation capabilities, and to allow the user to peacefully develop a portion of a non-gaming universe without being forced to strive towards a pre-defined goal.

AIML

AIML, or Artificial Intelligence Markup Language, is an XML dialect for creating natural language software agents.

The XML dialect called AIML was developed by Richard Wallace and a worldwide free software community between the years of 1995 and 2002. It formed the basis for what was initially a highly extended Eliza called "A.L.I.C.E." ("Artificial Linguistic Internet Computer Entity"), which won the

annual Loebner Prize Contest for Most Human Computer three times, and was also the Chatterbox Challenge Champion in 2004.

Because the A.L.I.C.E. AIML set was released under the GNU GPL, and because most AIML interpreters are offered under a free or open source license, many "Alicebot clones" have been created based upon the original implementation of the program and its AIML knowledge base. Free AIML sets in several languages have been developed and made available by the user community. There are AIML interpreters available in Java, Ruby, Python, C++, C#, Pascal, and other languages.

A semi-formal specification and a W3C XML Schema for AIML are available.

AIML (Artificial Intelligence Markup Language) is an XML-compliant language that's easy to learn, and makes it possible for you to begin customizing an Alicebot or creating one from scratch within minutes.

The most important units of AIML are:

<aiml>: the tag that begins and ends an AIML document

<category>: the tag that marks a "unit of knowledge" in an Alicebot's knowledge base

<pattern>: used to contain a simple pattern that matches what a user may say or type to an Alicebot

<template>: contains the response to a user input

There are also 20 or so additional more tags often found in AIML files, and it's possible to create your own so-called "custom predicates". Right now, a beginner's guide to AIML can be found in the AIML Primer.

The free A.L.I.C.E. AIML includes a knowledge base of approximately 41,000 categories. Here's an example of one of them:

<category>

<pattern>WHAT ARE YOU</pattern>

<template>

<think><set name="topic">Me</set></think>

I am the latest result in artificial intelligence,

which can reproduce the capabilities of the human brain

with greater speed and accuracy.

</template>

</category>

(The opening and closing <aiml> tags are not shown here, because this is an excerpt from the middle of a document.)

Everything between <category> and </category> is -- you guessed it -- a category. A category can have one pattern and one template. (It can also contain a <that> tag, but we won't get into that here.)

The pattern shown will match *only* the exact phrase "what are you" (capitalization is ignored).

But it's possible that this category may be invoked by another category, using the

<srai> tag (not shown) and the principle of reductionism.

In any case, if this category is called, it will produce the response "I am the latest result in artificial intelligence..." shown above. In addition, it will do something else interesting. Using the <think> tag, which causes Alicebot to perform whatever it contains but hide the result from the user, the Alicebot engine will set the "topic" in its memory to "Me". This allows any categories elsewhere with an explicit "topic" value of "ME" to match better than categories with the same patterns that are not given an explicit topic. This illustrates one mechanism whereby a botmaster can exercise precise control over a conversational flow.

Categories

Categories in AIML are the fundamental unit of knowledge. A category consists of at least two further elements: the pattern and template elements. Here is a simple category:

<category>

<pattern>WHAT IS YOUR NAME</pattern>

<template>My name is John.</template>

</category>

When this category is loaded, an AIML bot will respond to the input "What is your name" with the response "My name is John."

Patterns

A pattern is a string of characters intended to match one or more user inputs. A literal pattern like

WHAT IS YOUR NAME

will match only one input, ignoring case: "what is your name". But patterns may also contain wildcards, which match one or more words. A pattern like

WHAT IS YOUR *

will match an infinite number of inputs, including "what is your name", "what is your shoe size", "what is your purpose in life", etc.

The AIML pattern syntax is a very simple pattern language, substantially less complex than regular expressions and as such not even of level 3 in the Chomsky hierarchy. To compensate for the simple pattern matching capabilities, AIML interpreters can provide preprocessing functions to expand abbreviations, remove misspellings, etc.

Template

A template specifies the response to a matched pattern. A template may be as simple as some literal text, like

My name is John.

A template may use variables, such as the example

My name is <bot name="name"/>.

which will substitute the bot's name into the sentence, or

You told me you are <get name="user-age"/> years old.

which will substitute the user's age (if known) into the sentence.

Template elements include basic text formatting, conditional response (if-then/else), and random responses.

Templates may also redirect to other patterns, using an element called srai. This can be used to implement synonymy, as in this example (where CDATA is used to avoid the need for XML escaping):

```
<category>
<pattern>WHAT IS YOUR NAME</pattern>
  <template><![CDATA[My name is <bot name="name"/>.]]></template>
</category>
<category>
<pattern>WHAT ARE YOU CALLED</pattern>
<template>
  <srai>what is your name</srai>
</template>
</category>
```

The first category simply answers an input "what is your name" with a statement of the bot's name. The second category, however, says that the input "what are you called" should be redirected to the category that matches the input "what is your name"--in other words, it is saying that the two phrases are equivalent.

Templates can contain other types of content, which may be processed by whatever user interface the bot is talking through. So, for example, a template

may use HTML tags for formatting, which can be ignored by clients that don't support HTML.

AIML 1.0 Tag Set

<aiml>
<bot name="name"/>
<bot name="XXX"/>
<that index="2,1"/>
<that index="X,Y"/>
<that>
<category>
<input index="2"/>
<input index="3"/>
<condition name="X" value="Y">
<condition name="X">
<condition>
<gender>
<date/>
<id/>
<get name="name"/>
<size/>
<star index="X"/>
<thatstar index="X"/>
<get name="topic"/>
<topicstar index="X"/>
<version/>
<get name="XXX"/>
<gossip src="X">
<learn>X</learn>
<li name="X" value="Y">

<li value="Y">

<pattern>
<person/>
<person2>
<person2/>
<person>
<random>
<set name="topic">
<set name="name">
<set name="XXX">
<sr/>
<srai>
<system>
<template>
<think>
<topic name="X">
<uppercase>
<lowercase>
<sentence>
<formal>
<if name="X" value=Y">
<else>
<javascript>