An Introduction To Lambda Calculi For Computer Scientists

Lambda calculus

meaning. Hankin, Chris, An Introduction to Lambda Calculi for Computer Scientists, ISBN 0954300653 Monographs/textbooks for graduate students Sørensen

In mathematical logic, the lambda calculus (also written as λ-calculus) is a formal system for expressing computation based on function abstraction and application using variable binding and substitution. Untyped lambda calculus, the topic of this article, is a universal machine, i.e. a model of computation that can be used to simulate any Turing machine (and vice versa). It was introduced by the mathematician Alonzo Church in the 1930s as part of his research into the foundations of mathematics. In 1936, Church found a formulation which was logically consistent, and documented it in 1940.

== Definition ==

Alonzo Church

mathematical logic and the foundations of theoretical computer science. He is best known for the lambda calculus, the Church–Turing thesis, proving the unsolvability

Alonzo Church (June 14, 1903 – August 11, 1995) was an American computer scientist, mathematician, logician, and philosopher who made major contributions to mathematical logic and the foundations of theoretical computer science. He is best known for the lambda calculus, the Church–Turing thesis, proving the unsolvability of the Entscheidungsproblem ("decision problem"), the Frege–Church ontology, and the Church–Rosser theorem. Alongside his doctoral student Alan Turing, Church is considered one of the founders of computer science.

== Career ==

Alonzo Church was born on June 14, 1903, in Washington, D.C., where his father, Samuel Robbins Church, was a justice of the peace and the judge of the Municipal Court for the District of Columbia. He was the grandson of Alonzo Webster Church (1829-1909), United States Senate Librarian from 1881 to 1901, and great-grandson of Alonzo Church, a professor of Mathematics and Astronomy and 6th President of the University of Georgia. As a young boy, Church was partially blinded by an air gun accident. The family later moved to Virginia after his father lost his position at the university because of failing eyesight. With help from his uncle, also named...

Programming language theory

allowing computer programs to be expressed as mathematical logic. A team of scientists at Xerox PARC led by Alan Kay develop Smalltalk, an object-oriented

Programming language theory (PLT) is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of formal languages known as programming languages. Programming language theory is closely related to other fields including linguistics, mathematics, and software engineering.

== History ==

List of computer scientists

This is a list of computer scientists, people who do work in computer science, in particular researchers and authors. Some persons notable as programmers

This is a list of computer scientists, people who do work in computer science, in particular researchers and authors.

Some persons notable as programmers are included here because they work in research as well as program. A few of these people pre-date the invention of the digital computer; they are now regarded as computer scientists because their work can be seen as leading to the invention of the computer. Others are mathematicians whose work falls within what would now be called theoretical computer science, such as complexity theory and algorithmic information theory.

== A ==

Wil van der Aalst – business process management, process mining, Petri nets

Scott Aaronson – quantum computing and complexity theory

Rediet Abebe – algorithms, artificial intelligence

Hal Abelson – intersection of computing and teaching

Serge Abiteboul – database theory

Samson Abramsky – game semantics

Leonard Adleman – RSA, DNA computing

Manindra Agrawal – polynomial-time primality testing

Luis von Ahn – human-based computation

Alfred Aho – compilers book, the 'a' in AWK

Frances E. Allen – compiler optimization

Gene Amdahl – supercomputer developer, Amdahl Corporation founder

David P. Anderson – volunteer...

Conor McBride

Typed Lambda Calculi and Applications: 16–30. – (2002). "Elimination with a Motive" (PDF). Types for Proofs and Programs. Lecture Notes in Computer Science

Conor Titania McBride (born 18 February 1973) is a reader in the department of Computer and Information Sciences at the University of Strathclyde. In 1999, they completed a Doctor of Philosophy (Ph.D.) in Dependently Typed Functional Programs and their Proofs at the University of Edinburgh for their work in type theory. They formerly worked at Durham University and briefly at Royal Holloway, University of London before joining the academic staff at the University of Strathclyde.

They were involved with developing international standards in programming and informatics, as a member of the International Federation for Information Processing (IFIP) IFIP Working Group 2.1 on Algorithmic Languages and Calculi, which specified, maintains, and supports the programming languages ALGOL 60 and ALGOL 68.

They favor and often use the language Haskell.

== Research ==

Their most notable research is in the field of type theory. They cocreated the programming language Epigram with James McKinna. Several of their articles, including the joint-written article defining the Epigram language, have been published in the Journal of Functional Programming. They are also known for introducing applicative functors...

Curry–Howard correspondence

research is usually referred to as modern type theory. Such typed lambda calculi derived from the Curry–Howard paradigm led to software like Rocq in which

In programming language theory and proof theory, the Curry–Howard correspondence is a direct relationship between computer programs and mathematical proofs. It is also known as the Curry–Howard isomorphism or equivalence, or the proofs-as-programs and propositions- or formulae-as-types interpretation.

It is a generalization of a syntactic analogy between systems of formal logic and computational calculi that was first discovered by the American mathematician Haskell Curry and the logician William Alvin Howard. It is the link between logic and computation that is usually attributed to Curry and Howard, although the idea is related to the operational interpretation of intuitionistic logic given in various formulations by L. E. J. Brouwer, Arend Heyting and Andrey Kolmogorov (see Brouwer–Heyting–Kolmogorov interpretation) and Stephen Kleene (see Realizability). The relationship has been extended to include category theory as the three-way Curry–Howard–Lambek correspondence.

== Origin, scope, and consequences ==

The beginnings of the Curry–Howard correspondence lie in several observations:

Carl Hewitt

December 12, 1944 – December 7, 2022) was an American computer scientist who designed the Planner programming language for automated planning and the actor model

Carl Eddie Hewitt (; December 12, 1944 – December 7, 2022) was an American computer scientist who designed the Planner programming language for automated planning and the actor model of concurrent computation, which have been influential in the development of logic, functional and object-oriented programming. Planner was the first programming language based on procedural plans invoked using pattern-directed invocation from assertions and goals. The actor model influenced the development of the Scheme programming language, the π-calculus, and served as an inspiration for several other programming languages.

== Education and career ==

Hewitt obtained his PhD in mathematics at MIT in 1971, under the supervision of Seymour Papert, Marvin Minsky, and Mike Paterson. He began his employment at MIT that year, and retired from the faculty of the MIT Department of Electrical Engineering and Computer Science during the 1999–2000 school year. He became emeritus in the department in 2000. Among the doctoral students that Hewitt supervised during his time at MIT are Gul Agha, Henry Baker, William Clinger, Irene Greif, and Akinori

Yonezawa.

From September 1989 to August 1990, Hewitt was the IBM...

Turing completeness

until it is Turing-complete. The untyped lambda calculus is Turing-complete, but many typed lambda calculi, including System F, are not. The value of

In computability theory, a system of data-manipulation rules (such as a model of computation, a computer's instruction set, a programming language, or a cellular automaton) is said to be Turing-complete or computationally universal if it can be used to simulate any Turing machine (devised by English mathematician and computer scientist Alan Turing). This means that this system is able to recognize or decode other data-manipulation rule sets. Turing completeness is used as a way to express the power of such a data-manipulation rule set. Virtually all programming languages today are Turing-complete.

A related concept is that of Turing equivalence – two computers P and Q are called equivalent if P can simulate Q and Q can simulate P. The Church–Turing thesis conjectures that any function whose values can be computed by an algorithm can be computed by a Turing machine, and therefore that if any real-world computer can simulate a Turing machine, it is Turing equivalent to a Turing machine. A universal Turing machine can be used to simulate any Turing machine and by extension the purely computational aspects of any possible real-world computer.

To show that something is Turing-complete, it...

Natural deduction

comprehensive summary of natural deduction calculi, and transported much of Gentzen's work with sequent calculi into the natural deduction framework. His

In logic and proof theory, natural deduction is a kind of proof calculus in which logical reasoning is expressed by inference rules closely related to the "natural" way of reasoning. This contrasts with Hilbert-style systems, which instead use axioms as much as possible to express the logical laws of deductive reasoning.

== History ==

Natural deduction grew out of a context of dissatisfaction with the axiomatizations of deductive reasoning common to the systems of Hilbert, Frege, and Russell (see, e.g., Hilbert system). Such axiomatizations were most famously used by Russell and Whitehead in their mathematical treatise Principia Mathematica. Spurred on by a series of seminars in Poland in 1926 by Łukasiewicz that advocated a more natural treatment of logic, Jaśkowski made the earliest attempts at defining a more natural deduction, first in 1929 using a diagrammatic notation, and later updating his proposal in a sequence of papers in 1934 and 1935. His proposals led to different notations such as Fitch notation or Suppes' method, for which Lemmon gave a variant now known as Suppes–Lemmon notation.

Natural deduction in its modern form was independently proposed by the German mathematician...

Type theory

University Press. ISBN 978-0-521-05422-5. A good introduction to simple type theory for computer scientists; the system described is not exactly Church's

In mathematical logic, and theoretical computer science, type theory is the study of formal systems that classify expressions or mathematical objects by their types. Roughly speaking, a type plays a similar role to that played by a data type in programming: it specifies what kind of thing an expression is and how it may be used. Type theories are used in the study of programming languages (type systems), formal logic, and the formalization of mathematics.

Some type theories have been proposed as alternatives to set theory as a foundation of mathematics. Examples include Alonzo Church's simple theory of types and Per Martin-Löf's intuitionistic type theory.

Many proof assistants are based on type theory. For example, the underlying formal language of Rocq (formerly Coq) is the calculus of inductive constructions, while Lean is based on dependent type theory.

== History ==

https://uk.fulldoc.me/journal/478N4341Q2/976N21Q/mitsubishi-space__star_workshop_repair-manual-download-1998__2005.pdf
https://uk.fulldoc.me/journal/fs/1F5805S/cavalier__vending__service__manual.pdf
https://uk.fulldoc.me/lib/lu/U99L199499260916/reinforced-concrete-structures_design-according__to-csa.pdf
https://uk.fulldoc.me/ref/wh/5808H0W/kisah__wali-wali__allah.pdf
https://uk.fulldoc.me/slide/yt162609/Y9439T3652223939/maco-8000_manual.pdf
https://uk.fulldoc.me/ebook/fi162609/5477F21I03223939/panasonic_vdr_d210__d220_d230-series__service__manual_repair__guide panasonic-vdr_d100_d150__d152__d158_service-manual__repair-guide.pdf
https://uk.fulldoc.me/doc/70659TU/223939160926/evans-methods__in__psychological__research_2-edition__field__discovering_statistics_using__spss__3__e.pdf
https://uk.fulldoc.me/journal/E65P809937/E16P778/audi-a5_cabriolet_owners_manual.pdf
https://uk.fulldoc.me/slide/jl/3199L7J347260916/2006__international-building_code__structuralseismic__design_manual_volume-2_building-design-examples-for_lightframe-tiltup-and_masonry.pdf

https://uk.fulldoc.me/slide/D64R650/160926/mechanical__engineering-workshop_layout.pdf