Berühmte Informatiker

Teil 12:
Alonzo Church 1903 - 1995
John McCarthy 1927 -

Alonzo Church

- * 14.06.1903 in Washington
- † 11.08.1995 in Hudson

- 1924 Bachelor an der Princeton University
- 1927 Doktorarbeit an der Princeton University
- 1929-1967 Professor für Mathematik in Princeton
- 1967-1990 Professur an der University of California

Beiträge:

- Entwurf des λ-Kalküls
- Church-Turing-These

λ–Kalkül

- Das λ-Kalkül ist ein minimaler Kalkül, das den Kern aller funktionaler Programmier-sprachen modelliert.
- Das λ-Kalkül besteht nur aus zwei Bausteinen:
  - Funktionsdefinition (*Abstraktion*):

    $$\lambda\, x.A$$

    Definition einer anonymen Funktion mit dem Parameter *x* und einem Anweisungsblock *A*. Beispiel: die Identitätsfunktion f(*x*)=*x* wird im λ-Kalkül zu

    $$\lambda\, x.x$$

  - Funktionsanwendung (*Applikation*):

    $$F\ A$$

    Die Anwendung der Funktion F auf den Ausdruck A. Beispiel:

    $$(\lambda\, x.x)\ y \rightarrow y$$

- Die fundamentale Eigenschaft des λ-Kalküls ist, daß es nur Funktionsdefinition und Funktionsapplikation gibt. Es gibt keine Zahlen, Funktionsnamen, arithmetische Ausdrücke, Wahrheitswerte.
- Typisierte Varianten führten zur Entwicklung von Lisp, und (viel später) ML und Haskell
- Unter http://www.cip.physik.uni-muenchen.de/~tf/lambda/ findet man eine ausführliche Einführung in das λ-Kalkül, empfehlenswert ist auch http://www.inf.fu-berlin.de/lehre/WS03/alpi/lambda.pdf

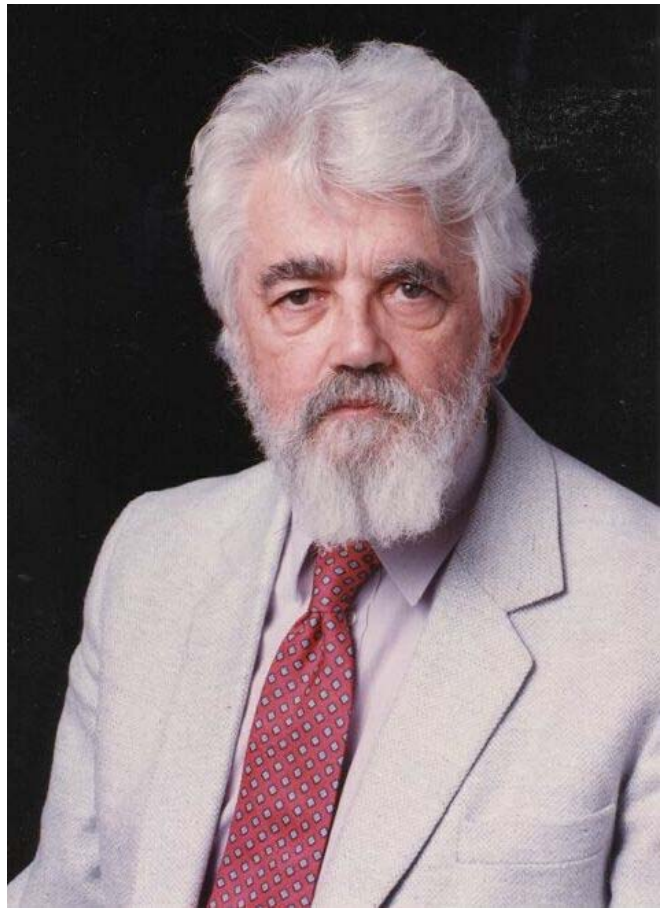
Church-Turing-These

- Die Church-Turing-These trifft eine Aussage über die Fähigkeiten einer Rechenmaschine. Sie lautet:

  Die Klasse der Turing-berechenbaren Funktionen ist genau die Klasse der intuitiv berechenbaren Funktionen.
- Unter der Klasse der intuitiv berechenbaren Funktion werden alle Funktionen verstanden, die prinzipiell auch durch einen Menschen ausgerechnet werden können.
- Church und Turing konnten des Weiteren zeigen, daß der λ-Kalkül die gleichen Probleme berechnen kann wie die Turing-Maschine (und umgekehrt).
- Mit der Church-Turing-These, von der man in der Informatik ausgeht, daß sie wahr ist, kann gezeigt werden, daß es verschiedene Probleme gibt, die nicht berechenbar sind. Beispiele dafür sind das *Halteproblem* oder das *Wortproblem*.

John McCarthy

* 04.09.1927 in Boston



- Logiker und Informatiker
- 1948 Bachelor of Science in Mathematik an California Institute of Technology
- 1951 Doktortitel an der Princeton University
- Heute: Emeritierter Professor der Stanford University

Beiträge:

- Prägung des Begriffs "Künstliche Intelligenz"
- Erfinder der Programmiersprache LISP
- Entwurf des Alpha-Beta-Algorithmus (Anwendung in der Spieltheorie)
- Erster mark&sweep Algorithmus zur automatischen Speicherbereinigung (garbage collection)

Auszeichnungen:

- Turing-Preis für seine Beiträge im Bereich künstlicher Intelligenz

LISP (**LIS**t **P**rocessor)

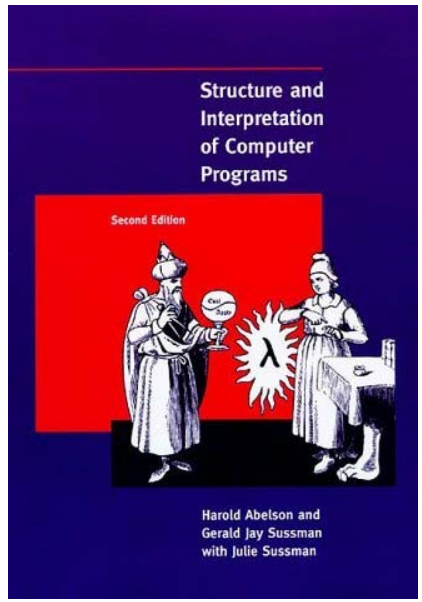
- LISP entstand als Implementierung des λ-Kalküls
- LISP besitzt als Grunddatenstrukturen *Skalar*werte, die *Atome* genannt werden und *Listen*.
- Listen können beliebig verschachtelt sein (Listen von Listen) und werden durch runde Klammern notiert.
- Programme in LISP können sowohl interpretiert, als auch durch einen Compiler in effizienten Maschinencode übersetzt werden.
- Es gibt ein "eval"-Konstrukt, mit Hilfe dessen sich ein als Parameter übergebenes Lisp-Programm ausführen läßt
- Der Lisp-Interpreter ist in seiner ursprünglichen Variante ein zwei Seiten langes Lisp-Programm!
- Beispiel für ein LISP-Programm:

```
(defun ! (n)
      (cond ((= n 0) 1)
            ((> n 0) (* n (! (- n 1))))))
```

  Aufruf:

```
(! 6)
```

  Welches Programm wird hier implementiert?
- Das Originalpaper zu LISP findet man unter http://www-formal.stanford.edu/jmc/recursive.html
- Implementierungen: **Lisp 1.5** (mit dynamic binding) und **Scheme** (static binding) (Verwendugn in Abelson/Sussman)
- Scheme war viele Jahre lang die Sprache für die Informatik-Ausbildung in den USA, jetzt zunehmend andere funktionale Sprachen



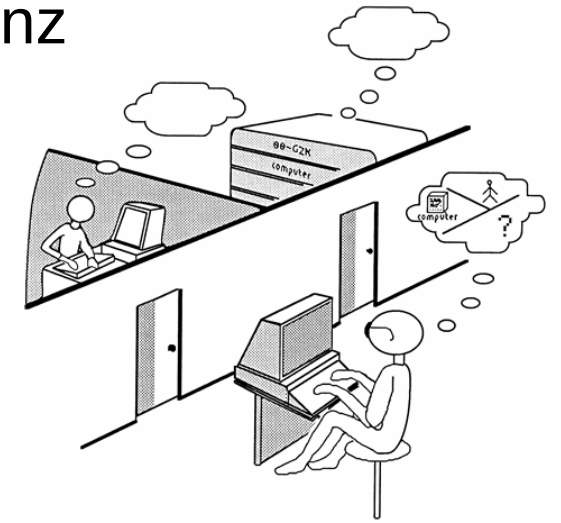
Garbage Collection

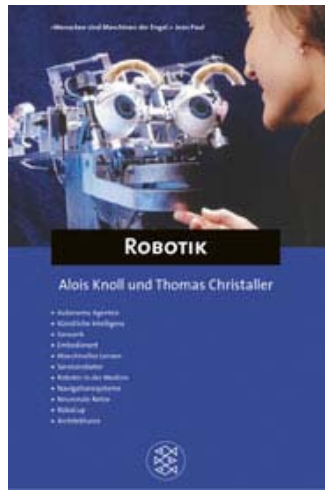
- LISP verfügte als eine der ersten Programmiersprachen über automatische "Müllabfuhr " (Garbage Collection).
- Problemstellung: Im Laufe einer Programmausführung wird laufend Speicher reserviert. Der Müllabfuhr-Mechanismus gibt den Speicher wieder frei, falls er nicht mehr benötigt wird.
- Es existieren drei verschiedene Ansätze:
  - Mark&Sweep: Ausgehend von als benutzt geltenden Objekten werden alle referenzierten Objekte markiert. Alle unmarkierten Objekte werden anschließend gelöscht.
  - Mark&Compact: Wie bei Mark&Sweep werden die Objekte markiert jedoch zusätzlich noch in einen „lebenden“ Speicherbereich kopiert. Der Vorteil: es entsteht ein großer, zusammenhängender freier Speicherbereich. Nachteil: Referenzen auf Objekte müssen aktualisiert werden $\Rightarrow$ fehleranfällig.
  - Referenzzählung: Für jedes Objekt wird die Anzahl der Referenzen verwaltet. Fällt die Anzahl auf 0, so kann der Speicherplatz freigegeben werden. Problem: Objekte, die sich gegenseitig als Referenz enthalten, aber nicht von anderen Objekten benutzt werden, z.B. doppelt verkettete Liste.

Künstliche Intelligenz

- Fachdisziplin der Informatik mit interdisziplinären Charakter
- Ziel: Maschinen sollen sich so menschlich wie möglich verhalten.
- Zum Messen der künstlichen Intelligenz wurde der Turing-Test entwickelt:

  Ein Mensch stellt an einem Terminal (ohne Sicht- und Hörkontakt) einem Menschen und einer Maschine Fragen. Kann der Mensch nicht unterscheiden, ob der Gesprächspartner Mensch oder Maschine ist, so ist die Maschine intelligent. Siehe auch die Ausführungen zum "Chinesischen Zimmer" in Knoll/Christaller
- Die künstliche Intelligenz gliedert sich in viele Einzelbereiche u.a. Spieltheorie (z.B. Schachcomputer), Dialogsysteme, Robotik.





Alpha-Beta-Algorithmus

- Der Alpha-Beta-Algorithmus ist ein Algorithmus zur Ermittlung der optimalen Spielstrategie für Spiele, bei denen zwei gegnerische Spieler abwechselnd Züge ausführen und optimiert den Mini-Max-Algorithmus.
- Grundidee des Minimax-Algorithmus:
  - Durchsuchen des vollständigen Spielbaums (der Baum, der von der aktuellen Spielsituation ausgehend alle möglichen Spielzüge enthält).
  - Die Blätter werden nach dem Spielausgang bewertet (maximaler Wert für Sieg, minimaler für Niederlage).
  - Ausgehend von den Blättern werden nun die einzelnen Knoten bewertet. Dabei wird immer davon ausgegangen, dass der jeweilige Spieler den für sich besten Ast wählt (der Spieler wählt also das Maximum der Nachfolgerknoten, der Gegner jeweils das Minimum).
  - **Problem**: der Spielbaum kann extrem groß und tief werden, eine komplette Auswertung ist für gängige Spiele nicht möglich.
- Der Alpha-Beta-Algorithmus liefert nun eine Strategie wie die Suche beschränkt werden kann, indem nur noch Pfade betrachtet werden, die das Ergebnis verbessern können.
- Eine ausführliche Erklärung des Minimax und Alpha-Beta-Algorithmus findet man in E. Rich "Aritficial Intelligence" oder ttp://www.kbs.unihannover.de/praktikum/aufgaben97/cppaufgabe/minimax.html