EI Zusammenfassung

Computational Thinking

Software Engineering:

- Prozessen und Techniken

- softwareprodukte entwickeln

- innerhalb eines Budgets und Zeitlimit

- zufriedenstellend funktionierend und qualititative Bedingungen

Thiniking Tools:

Abstraktion:

- Komplizierte Probleme vereinfachen indem man Nutzlose Details ausblendet

Dekomposition und Separation:

- Man kann Aufgaben aufteilen indem man sie in verscchiedene unterprobleme aufteilt. Welche dann von verschiedenen Personen bearbeitet werden.

Parallel algorithms:

- Mann kann Aufgaben an mehrere Arbeiter geben, so arbeiten mehr am selben und in der selben Zeit kann mehr erledigt werden.

Redundanz:

- Replikation kritischer Elemente erhöht Wiederstandsfähigkeit

Prefetching:

- Hohlen was man braucht bevor mans braucht

Cacheing:

- Zwischenspeichern was man wieder bald wiederbraucht

Planning and optimization

- Analysieren verschiedener Szenarien

Piplinging:

- effizientes Nutzen teurer Ressourcen

concurency control:

- geteilte ressourcen können zu safety and livenes problems führen

safety problems:

- nicht böses sollte passieren

liveness problems :

- etwas gutes sollte passieren

Simulation :

- teure experimente können durch billigere simulationen ersetzt werden.

Rekursion:

- Definiere einer Funktion durch das verwenden der Funktion selber. Um Folgeglied berechnen zu können muss man das vorherige kenne.

Breath-firs search:

- Anstrengend, findet alle lösungen
- Findet shortest path zuerst

Dept-first search:

- Findet erste Lösung
- Kann sein dass es nicht shortest path findet

Modelling:

- Verschieden Modelle
- Prototy
- Data modelling
- UML
- Finit State model

Programming Languages:

Language :

- Set of sequences of symbols that we interpret to attribute meaning

Turing machine :

- Read's and writes tape of 0s and 1s
- Language it accepts is the set of strings that leave it in an accepting state

Formal language :

- Use a set of rules (alpha -> beta) to describe the structure of the language

Programming language :

- Language to instruct a computer to compute « stuff »

Compiler

- Parse, transform, analyze, optimize, generate maschine code ot of programming language

Programming is modeling!!!!

Programming languages in common:

- Statements
- Control consructs
- Kewords
- Comments
- Functions
- Variables
- Numbers, strings
- Expressions

Historical:

- Punch cards
- Babbage's analytical engine
- Eniac
- 1. Gen Machine code (Harvard Mark1)
- Subroutines
- 2. Gen assembler (symbolic names for humans)
- 3. Gen Fortran (highlevel languages are born) Formula Translator
- Algol (Recursion, Block Structure
- Lisp(Garbae collection
- Cobol modules
- Basic (interactiv programming for masses, very easy syntax)

- Jcl (invented scripting
- Planner and Prolog (facts and rules, queries and inferences)
- Pascal (structured programming, successful with pc
- C (bridgin low and high-level programming, good for portable systems)
- Smalltalk (everything is an object, happens by sending messages)
- Bourne shell (scripting piplines of commands)
- SQL (domain-specific language for relational databases)
- Miranda (pure funtional programming, but lazy evaluation
- Perl, CGI (Text manipulation, then server side web scripting
- Java Script, AJAx (Client-side browser scripting

How do languages differ :

- Functional
- Object oriented
- Imperative
- Logic

Conclusion :

- Programming is modeling
- Programming languages have always evolved to bring programming closer tot he users problem
- We are still very early in the history of programming

Endliche Automaten:

Endliche Autormat:

- Ist ein 6-Tupel
- A = (i,O,Q,delta, q0, F
- I : Input Alphabet aus endlich vielen Zeichen
- O: Output Alphabet aus endlich vielen Zeichen
- Q: Zustandsmenge aus ‘’
- Delta: Übergangsfunktion
- Q0: Anfangszustand
- F: Menge der Endzustände(F ist teilmenge von Q)

Definition

Eine **Turing-Maschine** ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, E)$
wobei

- Q ist eine endliche Menge von Zuständen
- $\Sigma = \{0, 1\}$ ist das Eingabealphabet
- Γ ist das Arbeitsalphabet und $\Sigma \subset \Gamma$
- $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, N\}$ ist die Überführungsfunktion
- q_0 ist der Startzustand
- $\square \in \Gamma \setminus \Sigma$ ist das Blank
- $E \subseteq Q$ ist die Menge der Endzustände

Nicht-deterministisch

- falls für einen Zustand q und ein Inputzeichen x mehrere Folgezustände g' möglich sind

deterministisch

- der deterministsche Automat benötigt mehr Zustände
- zu jedem nicht-deterministischen Automaten gibt es einen deterministischen Automaten, welcher dieselbe Sprache akzeptiert

Reguläre Ausdrücke

- ein Alphabet ist eine nicht-leere, endliche Menge von Zeichen.
- Mit Sprache darstellbar

Theorem

- Eine Sprache heisst regulär genau dann, wenn sie von einem regulären Ausdruck beschrieben wird.
- Eine Sprache ist regulär genau dann, wenn sie von einem endlichen Automaten akzeptiert wird.

Grammatik:

- Ist ein 4-Tupel (N,T,S,P)
- N: endliche Menge von non.terminal Symbolen
- T: endliche Menge von terminal Symbbolen
- S element N: Startsymbol
- P: endlcihe Menge von Produktionen

Reguläre Grammatik

- Eine Grammatik heisst regulär falls alle Produktionen die Form n -> w haben, wobei:
- N : non-terminal Symbol
- W : ist das leere Wort oder ein Wort mit höchstens einem non-terminal Zeichen, welches am Schluss stehen muss.

Satz:

- Jede reguläre Grammatik erzeugt eine reguläre Sprache
- Zu jeder regulären Sprache gibt es eine reguläre Grammatik welche sie erzeugt.

Berechenbarkeit und Komplexität

Wichtige Fragen:

- Welche Pobleme sind algorithmisch lösbar
- Gibt es unlösbare Probleme

Antworten:

- Ein Problem ist eine Menge von Elementen
- Ein Algorithmus ist eine endliche Beschreibung einer eindeutigen Handlungsvorschrift zur Lösung eines Problems.
- Ein Algorithmus löst ein Problem A, falls der Algorithmus für beliebigen Input x korrekt entscheidet ob x element A oder x nichtelement A

These von Church:

- Die Klasse der Turing -berechenbaren funktionen stimmt genau mit der Klasse der intuitiv berecenbaren Funktionen überein.

(Un-)Entscheidbarkeit

- Eine Menge P Teilmenge {0,1}* heisst entscheidbar, falls es eine Turing Maschine M gibt, die die charakteristische Funktion von P berechent.
- Eine Menge heiss unentscheidbar, falls sie nicht entscheidbar ist.
- Halteproblem ist Beispiel für unentscheidbar

Zsmfassung:

- Berechenbarkeitstheorie unterscuht die Grenzen der theoretiscchen Berechenbarkeit. Fokus insbesondere auf unentscheidbaren Problemen.
- Dazu werden die Begriffe Algorithmus, Problem usw mathematisch definiert und unterscuht
- Das mathematische Modell eines Algorithmus ist die Turning Maschine
- Es gibt auch andere Modelle, die aber äquivalend sind
- Es gibt Probleme die nicht entscheidbar sind.

Komplexitätstheorie:

- Untersucht die Grenzen der praktischen Berechenbarkeit
- Fokus liegt auf entscheidbaren Problemen
- Wie viele Ressourcen (Speicher, Laufzeit) braucht man um ein Problem zu lösen
- Hierarchie von entscheidbaren Problemen geordnet nach der benötigten Komplexitär zum Lösen eines Problems
- Diese Hierarchie besteht aus Komplexitätsklassen P, NP, EXP usw

Messbarkeit:

- Es gibt zwei grundlegende Arten Komplexität zu messen:
- Laufzeit: Wie viele Rechenschritte bötigt eine Turing Maschine, um ein Problem zu lösen
- Speicher wie viele Bandfelder benötigt eine Turing Makschine, um ein Problem zu lösen.

Datenbanken

Grundlegende Operationen CRUD:

- Create
- Read
- Update
- Delete

ACID Eigenschaften:

- Atomarität: Eine Transaktion wird entweder ganz oder gar nicht ausgeführt
- Konsistenz: Nach Ausführung ist Datenbestand konsistent
- Isolation: Keine gegenseitige Beeinflussung von Transaktionen
- Dauerhaftigkeit: Die Auswirkungen einer Transaktion müssen im Datenbestand dauerhaft bestehen bleiben

Data Privacy:

- Provable Privacy: vertrauliche Information kann nicht hergeleitet werden
- K-anonymität: Es gibt zu jeder vertraulichen Anfrage mindestens k mögliche Antworten
- Perfect Privacy: A priori Wahrscheinlichkeiten über vertrauliche Informationen werden nicht verändert
- Differential Privacy: Rauschen hinzufügen, so dass nicht auf einzelne Einträge geschlossen werden kann, aber statistische Auswertungen fast nicht beeinträchtigt werden

Digitale Nachhaltigkeit:

Chancen der Digitalisierung:

- Informationen einfach und schnell teilen
- Viele digitale Ressourcen frei verfügbar
- Neue Kommunikationskanäle
- Besser Vernetzung, mehr Bildung
- Vertiefte Erkenntnisse

Dualität Digitalisierung und Nachhaltigkeit

- Nachhaltige Digitalisierung: Digitalisierung als Mittel zum Zweck
- a.) Potential der Digitalisierung Nutzen um Ressourzen zu sparen (bsp: Video Konferenz)
- b.) Negative Konsequenzen der Digitalisierung minimieren (bsp: Green IT um Strom in Rechenzentren zu sparen
- Weitere Probleme: Rohstoffförderung, Herstellung, Elektro Schrott
- Digitale Nachhaltigkeit: Digitaliserung als Gegenstand der nachhaltigen Entwicklung
- a.) Potential der digitalisierung nutzen um freies Wissen zu verbreiten (bsp: Open Content wie Wikipedie)
- b.) Negative Konsequenzen der Digitalisierung minimieren (bsp: Opensource Software Nutzen um Abhängigkeit zu reduzieren

Voraussetzungen für digitale nachhaltigkeit:

- Ausgereiftheit: Verständlichkeit, Vollständigkeit, Korrektheit, Modularität, Integrität
- Transparente Strukturen: Technise Offenheit, Transparenz ermöglicht Kontrolle
- Semantische Daten: Semantische Informationen sind maschinenlesbar, Wissen besser absorbiert, weiterverarbeitet, interpretiert und weiterentwickelt werden.
- Vereilte Standorte: Peer-to-Peer, Abhängigkeit von einem einzigen Standort, langfristige Verfügbarkeit der digitalen Güter verbessert
- Freize Lizenz: Allen ist es erlaubt Gut zu nutzen und zu verändern, freie Zugänglichkeit
- Geteiltes Wissen:

- Viele Personen aus unterschiedlichen Organisationen, Wissens-Abhängigkeit, Beiträge von vielen
- Partizipations-Kultur: Alle kompetenten leute können an Weiteretwicklung des digitalen Gut, peer-review, Ökosystem um digiales Gut stellt Parizipations-Kulur sicher
- Faire Führungsstrukturen: dezentral organisiert, faire governance-przesse, Merokratie oder Demokratie
- Breit abgestützte Finanzierung: finanzierung erfolgt durch verschiedene Akteure, breit abgestütze Finanzierung
- Beitrag zur nachhaltigen Entwicklung: DG und desse Ökosystem leisten Beitrag zur nacchhaltigen Entwicklung.

Explizites Wissen:

- Quellcode, Dokumentation
- Handbücher, Anleitungen
- Daten, Berichte

Tazites Wissen:

- Erfahrung, Fähigkeiten, Skills
- Ansichten, Verständnis, Modelle
- Intuition, Ideen, Emotionen

Meritokratie:

- Anderes System wie Demokratie
- Je grösser der Beitrag des Individuum, umso grösser die Kontrolle

Parldigi:

- Open source software
- Open standards
- Open Data
- Open Access
- Open Content
- Open Internet
- Open Government

Data handling & visualizations

Formulas:

Randomisierte Algorithmen

Randomisierte Algorithmen:

- Computerprogramme sind deterministisch

- Eine Programm ist eine Funktion
- Zufällige Schritte müssen extra eingebaut werden

Anwendung:

- Simulation
- Optimierung
- Kryptographie
- Engineering

Monte-Carlo-Algorithmus

- Terminiert immer
- Liefert möglicherweise ein falsches Resultat

Las-Vegas-Algorithmus

- Terminiert möglicherweise nicht
- Liefert immer ein korrektes Resultat

Kryptographie

Kryptographie

- Kryptologie = Kryptographie + Kryptanalyse
- Algorithmen und Systeme sind öffentlich, nur Schlüssel bleiben geheim
- Open Source Implementationen

Discrete Optimization Methods

Discrete representation

- Graphs

Discrete Optimization Methods

- Dynamic programming
- Shortest Path:
    - Dijkstra
    - Bellman-Ford
    - Floyd-Warshall
- Max Flow

Context: Representation and optimization are relevant for :

- Data Structures and Algorithms
- Databases

- Automata and Formal Languages
- Computer Vision
- 3D Geometry Processing

Graphs :

- Discrete Structures that describe a finite set of éléments and their relations
- Complete Graph : all nodes are connectet with every other node
- Cycle: a Graph where all nodes on the outside are connected({v1,v2}{vn-1,vn}{vn,v1}
- Hypercube : is a n Cube consisting two copies of Qn-1
- Subgraph: Only a sub crowd of vertexes and nodes

Dynamic Programming

- Break Problems into subproblems
- Combine their solutions into solutions to larger problems

Dikstra's Algorithm

- Solves the single source shortest path problem
- Fastest knwon algorithmn
- Limitation : all weihts on the edges must be positive
- Input: weigted directet/undirected Graph

Bellman-Ford algorithm

- Solves the single source shortest path problem
- Negative weights are allowed
- If the graph contains a circle with negative weight, the shortest path does not exist
- Input : weighted directed/undirected Graphs

Floyd-Warshall algorithm

- Solves the multiple soruce shortest path problem
- It is assumed there is no negative cycle in the graph
- Input: weighted directed/undirected Graph

Maximum flow applications

- Traffic movement
- Hydraulic Systems
- Electircal circuits
- Computer Network reliability
- Distributed computing

Ford-Fulkerson algorithm

- Solves the maximum flow problem
- Limitation: it is assumed that the weights are integer values
- Input a flow network

Minimal cuts

- A cut is a node partition(S,T), where s is in S and t is in T
- Capacity(S,T) is the sum of weightts of the edges leaving S
- The objective is to delete the « best » set of edges from a flow lnetwork in ordert o separate the source formt the sink
- The minimal cut problem is finding the partition (S,T) with minimum capacity(S,T)

Modellierung und Simulation

Arten:

- Wissenschaftliche & technische Anwendugnen
- Soziale & politische
- Filme Computerspiele etc

Häufige Forderung:

- Verallgemeinerung
- Modell soll anwendbar sein auf neue Experimente
- Im Idealfall: Vorhersagbarkeit, Simulation stimmt mit neuen Messungen überein

Nutzen:

- Verständnis und Erklärung von Beobachtungen
- Vorhersage von Beobachtungen
- Unterstützung bei Konstruktion

Probleme:

- Qualität des Modells oft schwierig zu beurteilen
- Vor allem wenn Messungen von Experimenten aufwändig

Klassifikation:

- Deterministisch: Mehrfache Ausführung ergibt immer dasselbe Resultat
- Stochastisch: Mehrfache Ausführung bringt unterschiedliche Resultate
- Statisch: Zeit spielt keine Rolle
- Dynamisch: Zeitabhängig

Diskret

- Modell enthält Objekt aus einer abzählbaren Mebge
- Häufig Menge von Ereignissen

Stetig

- Modell enthält kontinuierliche Grössen
- Häufig Differentialgleichungen, stetige Funktionen

Diskretisierung zwei Komponente

Abtastung:

- Tabellierung einer endlichen Anzahl von Werten

Quantisierung:

- Digitale Approximation reeller Zahlen mit Gleitkomma Darstellung

4 Funktionsklassen