Informatik (als Nebenfach)
Vorlesung
Wintersemester 2003–2008, 2020

Johannes Waldmann, HTWK Leipzig

25. Januar 2022

Überblick

Informatik: was und warum?

- Informatik =
  Wissenschaft von (der Verarbeitung symbolisch repräsentierter Information durch) Algorithmen
  als Ingenieurdisziplin:
  Entwurf, Konstruktion, Betrieb von Softwaresystemen
- Ziele der VL für Studenten im Nebenfach:
  - effiziente Kommunikation mit Informatik-Fachleuten
  - Methoden zu Beschreibung technischer Systeme
  - Allgemeinbildung, insbesond. zur privatwirtschaftlichen und staatlichen Überwachung mittels IT-Systemen

Algorithmus und Programm

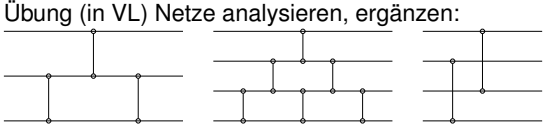
- Algorithmus — Beispiel (Alg. von Euklid, 300 v.Chr.)
  - Eingabe: zwei positive natürliche Zahlen.
  - Rechnung: solange beide verschieden sind: ersetze die größere durch ihre Differenz zur kleineren,
  - Ausgabe: eine der (dann gleichgroßen) Zahlen
- Algorithmus — Definition: eine in Schritte unterteilte exakte Rechenvorschrift, nach der für eine Klasse von Eingaben jeweils eine Ausgabe bestimmt wird.
- Programm — Definition: Realisierung eines Algorithmus in einer konkreten Programmiersprache zur Ausführung auf einer konkreten Rechenmaschine (Computer)
  ... oder mehreren (verteilter Algorithmus, Rechnernetz)

Zum Algorithmus von Euklid

- Algorithmus:
  - Eingabe: $x_0 > 0, y_0 > 0$
  - Rechnung: für $k = 0, 1, \ldots$:
    wenn $x_k > y_k$, dann $(x_{k+1}, y_{k+1}) = (x_k - y_k, y_k)$.
    wenn $x_k < y_k$, dann $(x_{k+1}, y_{k+1}) = (x_k, y_k - x_k)$.
  - Ausgabe: falls $x_k = y_k$, dann x_k.
- Eigenschaften:
  - für alle k: $x_k > 0, y_k > 0$
  - für alle $d \in \mathbb{N}, k \geq 0$: $(d|x_k \wedge d|y_k) \iff (d|x_{k+1} \wedge d|y_{k+1})$
  - für alle k: die Menge der gemeinsamen Teiler (gT) von x_0, y_0 ist genau die Menge der gT von x_k, y_k
  - die Ausgabe ist der größte gT der Eingabe

Sortiernetze

- als Beispiel für Rechnernetz
- jeder Rechner ist ein *Komparator*: Eingänge (links) x, y, Ausgänge (rechts): oben: $\max(x, y)$, unten: $\min(x, y)$
- ein Netz heißt *Sortiernetz*, wenn es zu jeder Eingabe eine monoton steigende Ausgabe berechnet
- Übung (in VL) Netze analysieren, ergänzen:



ist der letzte (rechte) Komparator im linken Netz nötig?
linkes/mittleres: wie geht das für Breite 5 und größer?
rechtes: wo sind min und max jetzt? wo sollen sie hin?

Gliederung dieser VL (Plan)

- Hardware (Informations- und Zahlendarstellungen, Rechenwerk, Speicher, Peripherie-Geräte, Betriebssysteme)
- Software (Datenmodellierung: Mengen, Funktionen, Relationen, Graphen, Bäume; Spezifikationen/Schnittstellen und Implementierungen, strukturierte imperative Programmierung)
- Rechnernetze (Adressierung, Protokolle, offen/geschlossen, föderiert/zentral; HTTP, E-Mail; Verschlüsselung, Signaturen; Anzeigenmarkt, Überwachungskapitalismus, informationelle Selbstbestimmung)

Quellen

- Skript, Organisatorisches: `https://informatik.htwk-leipzig.de/waldmann/lehre.html`
- Hartmut Ernst, Jochen Schmidt, Gerd Beneken: *Grundkurs Informatik*, Springer 2020 `https://link.springer.com/book/10.1007/978-3-658-30331-0`
- Heinz Peter Gumm, Manfred Sommer: *Einführung in die Informatik*, De Gruyter 2013, `https://www.degruyter.com/view/title/313768`
- Friedrich L. Bauer: *Kurze Geschichte der Informatik*, München 2007, `https://digi20.digitale-sammlungen.de//de/fs1/object/display/bsb00064605_00001.html`

- Uwe Kastens, Hans Kleine Büning: *Modellierung*, Hanser 2014, `http://www.hanser-elibrary.com/doi/book/10.3139/9783446442498`

Organisation dieser LV

- Veranstaltungen: jede Woche eine VL, eine Ü (Präsenz) und asynchrone Fernlehre: Diskussion in Forum
- Prüfungs(vor)leistungen:
  - PKB: online-Aufgaben (wöchentlich, individuell)
  - PP: Hausaufgaben-Präsentation
    (in Gruppen je ca. 4 Personen)
    in jeder Ü präsentieren 3 Gruppen, jeweils eine Aufgabe
    KW 42: 1A, 2A, 3A; KW 43: 4A, 5A, 6A;
    KW 44: 1B, 2B, 3B; KW 45: 4B, 5B, 6B; usw.
  - PK: Klausur
- E-Mail *ausschließlich von Hochschul-Adresse aus*

Hausaufgaben (ohne Wertung)

- (Ohne Wertung. Trotzdem sollte das jeder jetzt machen. Genauere Diskussion später in der VL, Abschnitt Computernetze.)

  stellen Sie den Browser datenschutzgerecht ein, vgl. `https://restoreprivacy.com/firefox-privacy/`.

  Für Video-Konferenzen (Fernlehre) wird doch WebRTC benötigt, legen Sie dafür in `about:profiles` ein neues Profil an, das Sie dann *nur* für Konferenzen (mit vertrauenswürdigen Gegenstellen) benutzen.

  Suchen Sie die Studienordnung Ihres Studienganges. Lesen Sie das Zertifikat des Webservers.

Welche Datenübertragungen finden bei Benutzung von Opal, autotool, Verlagswebseite für Ebook (von außerhalb des Hochschulnetzes) statt? Welche Rolle spielt die Trennung von Identitäts- und Service-Provider?

Genaueres dazu später im Abschnitt Computernetze.

- (kommt später dran, Sie können schon anfangen zu überlegen) In einem Behälter sind 91 Atome vom Typ A, 25 B, 4 C. Zwei unterschiedliche Atome können zu einem einzigen des dritten Typs reagieren. Nach einigen solchen Reaktionen gibt es nur noch Atome eines Typs. Welches? Wieviele höchstens?

  Geben Sie Reaktionsfolgen an, bei denen eine möglichst hohe Anzahl entsteht. Begründen Sie, daß keine höhere

Anzahl erreichbar ist.

Hinweise: benutzen Sie ...

- die Parität (gerade/ungerade) der Anzahlen,
- die Summe der Anzahlen B und C.

Beziehung zur LV (wird später deutlicher): *Invariante* und *Schrittfunktion* einer bedingungsgesteuerten Schleife

Hausaufgaben (zur Bewertung in KW 42)

1. Lesen Sie die angegebene Literatur im Browser, beginnen Sie mit der Suche im Katalog unserer Bibliothek.

   In diesem Buch von Gumm/Sommer werden Eigenschaften für einen „durchschnittlich ausgestatteten PC für unter 1000 EUR“ angegeben, die bei Drucklegung typisch waren.

   Vergleichen Sie mit PCs von heute (für ähnlichen Preis).

   Vergleichen Sie mit entsprechenden Angaben in früheren Ausgaben dieses Buches.

   Vergleichen Sie mit aktuellem Mobiltelefon.

   Vergleichen Sie mit dem Bordcomputer von Apollo 11.

Geben Sie für dessen Daten eine verläßliche Quelle an (möglichs eine primäre) (Wikipedia ist *niemals* Primärquelle, deswegen nicht zitierfähig, kann aber gelegentlich benutzt werden, um eine solche zu finden)

2. zum Algorithmus von Euklid:

   Das Verfahren vorführen (a) für Eingabe $30, 12$ (b) für Eingaben, die erst dann in der Ü genannt werden.

   Dabei die Eigenschaften $\mathsf{ggT}(x_0, y_0) = \mathsf{ggT}(x_i, y_i)$ überprüfen: jeweils die Menge der Teiler von x_i, von y_i, und die gemeinsamen Teiler angeben.

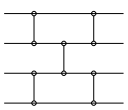
   Warum müssen die Eingaben positiv sein? (Was passiert sonst?)

   Geben Sie Eingaben an, für die größere Zahl in jedem

Schritt wechselt $(x_0, y_1, x_2, \dots)$, insgesamt genau 5 Subtraktionen stattfinden und das Resultat 1 ist.

(Beziehungen zur LV: 1. Analyse von Programmen mit Schleifen, 2. dieser Algorithmus gehört zum Kulturerbe der Menschheit, 3. wird zur Implementierung von Verschlüsselungs- und Signier-Verfahren benutzt: HTTPS, RSA, SSH)

3. Diskutieren Sie das Netz



   - die Rechnung durchführen für eine Eingabe (a) von Ihnen gewählt, (b) von mir gewählt.
   - warum ist das kein Sortiernetz? (eine Eingabe zeigen,

für die die Ausgabe nicht monoton ist)

   - das Netz reparieren durch Hinzufügen eines Komparators (die vorhandenen nicht ändern), die Korrektheit der Reparatur begründen (mit Methode aus der VL: wo sind min und max?)
   - verallgemeinern auf Breite 6, 8, ... $2n$. Wieviele Komparatoren?
     (Die Form läßt sich leicht verallgemeinern, der Korrektheitsbeweis ist aber schwierig—müssen Sie nicht unbedingt führen)

Die Überwachungswirtschaft

Überblick

- hier nur Überblick und elementare Schutzmaßnahmen
  z.B. für WWW-Recherchen für Lehrveranstaltungen
  (später technisch genauer, Abschnitt Computernetze)
- J. Waldmann: Einführung OS Überwachungskapitalismus (WS 2019) `https://imweb.imn.htwk-leipzig.de/~waldmann/talk/19/ubkap/`
- W. Hesse: Das Zerstörungspotential von Big Data und Künstlicher Intelligenz für die Demokratie, Informatik Spektrum 43(5) 2020, `https://link.springer.com/journal/287/volumes-and-issues/43-5`

Datenvermeidung beim Browsen

- Suche: `https://duckduckgo.com/`
  auch finanziert über Anzeigen, diese nur auf Such-Begriffe bezogen, nicht auf Benutzeridentifikation
- Browser: Firefox — auch finanziert durch Google, weil das als Default-Suchmaschine eingestellt ist
- Firefox datenschutzgerecht einstellen
  `https://restoreprivacy.com/firefox-privacy/`
  z.B. *search suggestions* übertragen Tastenanschläge in Echtzeit, Tipp-Geschwindigkeit und -fehler sind Indikator für Bildung, korreliert mit verfügbarem Einkommen
  Datensammlung durch Anzeigennetzwerke verhindern mit uMatrix, uBlock Origin, Multi-Account-Container

Sogenannte Content-Plattformen

- (3. Nov. 2020) `https://www.heise.de/news/TikTok-und-Sony-Music-Entertainment-vereinbaren-Zus.html` „Lizenzvereinbarung . . . soll die Verwendung der gesamten Musikbibliothek Sonys auf der Videoplattform ermöglichen.
- d. h., die Plattform zahlt (anstelle der „Autoren“) die Lizenzkosten, um mehr Konsumenten anzuziehen.
  Diese aber bitte auf der Plattform bleiben: nur dort kann man Daten sammeln, nur dafür sind Inhalte lizensiert.
- (23. Okt. 2020) „RIAA DMCAs GitHub into nuking popular YouTube video download tool“ `https://www.theregister.com/2020/10/23/riaa_youtube_dl_github/`

vgl. Danny O’Brien (2. Nov. 2020)
`https://www.eff.org/deeplinks/2020/11/github-youtube-dl-takedown-isnt-just-problem-americ`

Information und Daten

Überblick

- Computer verarbeitet Daten, die Information repräsentieren. (Gumm/Sommer 1.2, 1.3)
- Information gemessen in Bit, 1 Bit = 1 Ja/Nein-Frage, physikalisch (elektrisch) repräsentiert durch (z.B.) Strom (fließt oder nicht), Potential (hoch oder tief),
- im Computer gespeicherte und zwischen Computern transportierte Daten sind immer Folgen von Bits repräsentieren Zahlen, Buchstaben, Texte, Bilder, ...
- jeder Verarbeitung durch (heutige) Computer ist schließlich eine elektronische Verarbeitung vieler Bits

Zahlendarstellungen

Überblick

- Gumm/Sommer 1.4
- Zahlenbereiche in der Mathematik: natürliche ($\mathbb{N}$), ganze ($\mathbb{Z}$), rationale ($\mathbb{Q}$), reelle ($\mathbb{R}$), komplexe ($\mathbb{C}$), ...
- im Computer darstellbare Zahlenbereiche:
  - auf Hardware direkt zu verarbeiten (Maschinenzahlen): endliche Teilmengen von $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ Teilmenge endlich $\Rightarrow$ nicht jede Zahl darstellbar
  - durch Software zu verarbeiten: beliebig große, beliebig genaue Zahlen

Darstellung natürlicher Zahlen

- verwendet Positions-Systen mit Basis $B > 1$, Folge von Ziffern $(d_n, \ldots, d_1, d_0)_B$ mit $\forall i : 0 \leq d_i < B$ repräsentiert Zahl $\sum_{i=0}^{n} d_i \cdot B^i$
- Dezimalsystem (Zehnersystem): $(3, 1, 4)_{10}$ repräsentiert $3 \cdot 10^2 + 1 \cdot 10^1 + 4 \cdot 10^0 = 300 + 10 + 4 = 314$
- Binärsystem (Zweiersystem), im Computer verwendet: $(1, 1, 0, 1)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13$
- Darstellung einer Zahl n zur Basis B:
  - 0 dargestellt durch leere Ziffernfolge, sonst:
  - letzte Stelle (d_0) ist der Rest der Division von n durch B
  - Stellen davor $(d_n, \ldots, d_1)$ sind Darstellung von $\lfloor n/B \rfloor$

Rechnen in Positionssystemen

- ... wie in der Schule gelernt,
- Beispiel: Addition
  - beginnend bei niederwertigster Stelle, ggf. Überträge
  - Beispiel im Dezimalsystem: $314_{10} + 9817_{10}$
  - Beispiel im Binärsystem: $11010_2 + 1011_2$
- Beispiel: Multiplikation
  - Multiplikation mit einer Ziffer, Bsp: $134 \cdot 7$
  - Mult. beliebiger Zahlen, Bsp: $134 \cdot 752 = 134 \cdot (7 \cdot 10^2 + 5 \cdot 10^1 + 2 \cdot 10^0)$ $= 134 \cdot 7 \cdot 10^2 + 134 \cdot 5 \cdot 10^1 + 134 \cdot 2 \cdot 10^0$ Parallelogramm-Schema, spaltenweise Additionen
- Binär: genauso, Multiplikation ist sogar etwas einfacher

Maschinenzahlen (Binärzahlen fester Breite)

- Prozessor berechnet arithmetische Op. auf Binärzahlen bestimmter Breite (Stellenzahl) in einem Schritt
- Breite ist je nach Epoche verschieden, z.B. 6502 (1975) Apple II (1977): 8 Bit; AMD Opteron (2003): 64 Bit.
- und größere Zahlen sind nicht (direkt) darstellbar Ü: die größte mit 32 Bit darstellbare Zehnerpotenz?
- was soll ein 8-Bit-Rechner tun bei $20_{10} \cdot 20_{10}$?
  - anhalten (abstürzen)?, mit Unsinn weiterrechnen? beides gefährlich, muß durch geeignete Programmierung verhindert werden

Ganze Zahlen als Maschinenzahlen

- d.h., mit fixierter Breite w, Beispiele hier für $w = 4$
- als natürliche Zahlen sind darstellbar $0 \ldots 2^w - 1$
- Darstellung ganzer (d.h., vorzeichenbehafteter) Zahlen:
  - Maschinenzahl $0 \leq m < 2^{w-1}$ bezeichnet m
  - Maschinenzahl $2^{w-1} \leq m < 2^w$ bezeichnet $m - 2^w$. Bsp: $m = 13$, $2^3 \leq m < 2^4$, bezeichnet $13 - 2^4 = -3$.
- der Prozessor berechnet $(-3) + 5$ so:
  - Darstellungen (als natürliche Zahlen) addieren,
  - obersten Überlauf ignorieren.

  ist korrekt, denn $(2^4 - 3) + 5 = 13 + 5 = 18 = 2^4 + 2$

Vorzeichenwechsel

- für diese Festlegung ($2^{w-1} \leq m < 2^w$ bezeichnet $m - 2^w$)
- gegeben: n, gesucht die Darstellung von $-n$, Bsp: gegeben $n = 3 = (0011)_2$, gesucht $(1101)_2 = 2^4 - 3$
- Verfahren: komplementiere jede Ziffer ($0 \leftrightarrow 1$), addiere 1 Bsp: $0011 \to 1100 \to 1101$, Ü: dasselbe für $n = 4$
- Begründung: $(1-0, 1-0, 1-1, 1-1)_2 + 1 =$ $= 1111_2 - 0011_2 + 1 = (2^w - 1) - n + 1 = 2^w - n$
- diese Ganzzahl-Darstellung heißt *Zweierkomplement* (ohne die abschließende Addition: Einerkomplement)
- damit Subtraktion $x - y$ als Addition: $x + (-y)$ Beispiele: $5 - 3$, $3 - 5$.

Darstellung rationaler Zahlen

- Darstellung mit Komma (DE) oder „Dezimal“punkt (UK): $(d_n \ldots d_1 d_0, d_{-1} d_{-2} \ldots d_{-p})_B$ repräsentiert $\sum_{i=-p}^{n} d_i B^i$
- Bsp: $(10, 011)_2 = 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} =$ $2 + 0 + 0 + 1/4 + 1/8 = 19/8 = 2 + 3/8 = (2, 375)_{10}$
- Darstellung mit Exponent: $q = v \cdot m \cdot B^e$ mit
  - Vorzeichen $v \in \{-1, 0, +1\}$ (oder äquivalent)
  - Mantisse $m \in \mathbb{Q}_{>0}$ mit Komma, Exponent $e \in \mathbb{Z}$.

  Mantisse heißt *normiert*, wenn $1 \leq m < B$.

  Ü: was bedeutet das für $B = 2$? (das führende Bit ist ...)
- B: $(10.011)_2 \cdot 2^{(101)_2} = 19/8 \cdot 2^5 = 19/2^3 \cdot 2^5 = 19 \cdot 2^2 = 76$. mit normierter Mantisse: $(1.0011)_2 \cdot 2^{(110)_2} = 19/2^4 \cdot 2^6$

Darstellbare rationale Zahlen

- Gleitkommazahl $q = v \cdot m \cdot B^e$

  Bsp: $B = 2$, normierte Mantisse mit 3 Bit, Exponent (im Zweierkomplement) mit 3 Bit.
- darstellbar sind (Ü: wieviele Zahlen sind das?)
  - positiv: $1.00_2 \cdot 2^{-4} = 1/16 = 0.0625$, $1.01_2 \cdot 2^{-4} = 5/4/16 = 0.078125$, $\ldots 1.10_2 \cdot 2^3 = 3/2 \cdot 2^3 = 12$, $1.11_2 \cdot 2^3 = 7/4 \cdot 2^3 = 14$
  - 0
  - negativ: $-14, -12, \ldots, -0.078125, -0.0625$
- unterschiedliche Abstände, größere Lücke bei 0
- standardisiert (IEEE 754) und in Hardware implementiert: 32 Bit (24 Mantisse + 8 Exponent), 64 Bit (53 + 11)

Bestimmung der Gleitkomma-Darstellung

- Gleitkommazahl $q = v \cdot m \cdot B^e$

  Bsp: $B = 2$, normierte Mantisse mit 3 Bit, Exponent (im Zweierkomplement) mit 3 Bit.
- Bsp: gesucht: die zu $q = 2.7$ nächste darstellbare Zahl
  - Exponent e der nächstkleineren Zweierpotenz: $2^e \le q < 2^{e+1}$, also $q = m \cdot 2^e$ mit $m = 1.35, e = 1$
  - die Mantisse hat $w = 2$ (binäre) Nachkommastellen: $m \cdot 2^w$ runden, binär darstellen $1.35 \cdot 2^2 = 5.4 \approx 5 = (101)_2$, also $1.35 \approx (1.01)_2$
  - insgesamt $q = 1.35 \cdot 2^1 \approx (1.01)_2 \cdot 2^1 = 5/4 \cdot 2 = 2.5$
- Bsp: $q = 0.15$. Dann $2^{-3} < q < 2^{-2}$, $q = 1.2 \cdot 2^{-3}$,...

Fehlerquellen bei Gleitkomma-Rechnungen

- Standard IEEE-754: Prozessor-Arithmetik auf Gleitkomma-Maschinenzahlen liefert immer die zum exakten Resultat nächste darstellbare Zahl
- Bsp: $B = 2$, normierte Mantisse mit 3 Bit, Exponent (im Zweierkomplement) mit 3 Bit.
- $x = (1.10_2 \cdot 2^3), y = (1.00_2 \cdot 2^{-4})$, $x \oplus y = 12.0625 \approx 12 = x$ dann $(-x \oplus x) \oplus y = y$, aber $-x \oplus (x \oplus y) = -x \oplus x = 0$ d.h., Gleitkomma-Addition ist nicht assoziativ
- David Goldberg: *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, ACM Computing Surveys, 1991 `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.6768`

Zusammenfassung Zahlendarstellungen

- Darstellungen, Beispiele
  - natürliche Zahlen: Positionssystem, $(10011)_2 = (19)_{10}$
  - ganze Z.: Zweierkomplement $(10011)_2 = 19 - 2^5 = -21$
  - rationale Z., Festkomma: $(100.11)_2 = (10011)_2/2^2$
  - rationale Z., Gleitkomma: $(1.0011)_2 \cdot 2^{(10)_2}$
- Eigenschaften, typische Anwendungen
  - nat., ganz: exakt, z.B. für: Anzahlen
  - kleine ganze Z.: Maschinenzahlen (Hardware)
  - große nat. Z.: (Software) Verschlüsselung, Signatur
  - rat. Z. (Gleitkomma): genäherte Maßzahlen physikalischer Größen (Bsp: Audio-Signale)
  - ... feste Bitbreite: Hardware, variable: Software (MP3)

Hausaufgaben

1. wieviele Bit (in Text-Dokumenten) werden an der HTWK pro Semester insgesamt im Zusammenhang mit Lehrveranstaltungen erzeugt?

   Was kostet es, diese abzuspeichern (bei aktuellen Preisen für HDD oder SSD)? — Antwort: fast nichts, teuer ist nicht das Abspeichern, sonder das Herstellen (die Arbeitszeit)

   Geben Sie eine begründete Schätzung an, ausgehend von dieser LV. Ergänzen, diskutieren, verwenden Sie diese Schätzungen:

   - 8 Bit (= 1 Byte) pro Zeichen
   - Zeichen pro Zeile

   - Zeilen pro Folie
   - Folien pro Vorlesung
   - Vorlesungswochen im Semester
   - Vorlesungen je Professor und Semester
   - Professoren je Fakultät
   - Fakultäten

   Bestimmen Sie vernünftige untere und obere Schranken für jeden einzelnen Schätzwert und für das daraus berechnete Resultat.

2. für Binärzahlen der Breite 6:

   - stellen Sie dar: $5, 7$
   - berechnen Sie: $5 + 7, 5 \cdot 7$

   für Binärzahlen der Breite 6 im Zweierkomplement:

   - stellen Sie dar: $5, 11, 18, -18$,
   - berechnen Sie: $11 + 18$ und $11 - 18$,

   überprüfen Sie alle Resultate durch Rechnung im Dezimalsystem.

3. Die Maßzahl einer physikalischen Größe soll mit höchstens 0.5 Prozent relativem Fehler dargestellt werden.

   Wieviele Stellen muß die Mantisse der normierten binären Gleitkomma-Darstellung dafür wenigstens haben?

   Für normierte binäre Gleitkommazahlen mit 4 Bit für Mantisse, 4 Bit für Exponent:

   Nenne Sie die größte, kleinste, kleinste positive darstellbare Zahl.

   Wir bezeichnen mit $D(x)$ die zu x nächstliegende darstellbare Zahl.

   - bestimmen Sie $D(59), D(75)$.
   - welche Auswirkung hat jeweils die Änderung der letzten Stelle der Mantisse?
   - vergleichen Sie $D(75) - D(59)$ mit $D(75 - 59)$.

4. (Zusatz—ohne Wertung) Tragen Sie Ziffern 1 bis 5 ein, so daß ein korrektes Multiplikations-Schema entsteht:

```
        x x x x x
      . x x x x x
------------------
```

```
        x x x x x x
        x x x x x
    x x x x x x
    x x x x x
x x x x x
-----------------
x x x x x x x x x
```


Rechnen mit Wahrheitswerten

Einleitung, Motivation

- Bit = Wahrheitswert = Antwort auf Entscheidungsfrage
- Rechnen mit Bits wird in Hardware implementiert (elektronische Schaltungen in Prozessor)
- es gelten Rechengesetze ähnlich zum Zahlenrechnen
- beruht auf allgemeinen mathematischen Kulturtechniken: Term-Bäume, Term-Umformungen
- die auch sonst in der Informatik nützlich sind, z.B. Dokument-Bäume (DOM, XML),
- spezielle Anwendung: Rechnen mit Bildern (Bitmustern)
- Gumm/Sommer 5.2, vgl. Kastens/Kleine Büning 4.1

Wahrheitswerte und Funktionen

- $\mathbb{B}$ = Menge der Wahrheitswerte = {Falsch, Wahr}
  oft notiert als $\{0, 1\}$
- jede Funktion $f : \mathbb{B}^k \to \mathbb{B}$ heißt *aussagenlogische* oder *Boole*sche Funktion (Gumm/Sommer: Schaltfunktion)
  Bsp: die *Konjunktion* (Und-Verknüpfung) ist die Funktion $\{((0,0),0),((0,1),0),((1,0),0),((1,1),1)\}$
- George Boole (1815–1864): *An investigation into the Laws of Thought, on Which are founded the Mathematical Theories of Logic and Probabilities* (1854)
  `http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Boole.html`

Wertetabellen

- Für jedes $f : \mathbb{B}^k \to \mathbb{B}$ ist der Definitionsbereich endlich. (Wie groß genau? Wieviele Zeilen hat die Wertetabelle?)
- jedes solche f kann deswegen vollständig durch eine Wertetabelle (G/S: Schalttabelle) gegeben werden. Bsp:

x	y	$f(x,y)$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$g(x,y)$
0	0	0
0	1	0
1	0	0
1	1	1

x	$h(x)$
0	1
1	0

- viele Eigenschaften Boolescher Funktionen kann man durch komplette Fallunterscheidung feststellen,
  Bsp: für alle x, y gilt $h(f(x,y)) = g(h(x), h(y))$

Null- und einstellige Boolesche Funktionen

- nullstellig: die Konstanten `false` und `true`
- einstellig:
  - die identische Funktion

x	id(x)
0	0
1	1

  - die Negation (Verneinung, Nicht, $\neg$)

x	$\neg x$
0	1
1	0

  - es gibt zwei weitere einstellige Boolesche Funktionen, welche Werteverläufe haben diese, warum sind diese nicht so interessant?

Zwei- und mehrstellige Boolesche Funktionen

- zweistellig: Disjunktion (Alternative, Oder, $\vee$), Konjunktion (Und, $\wedge$), Implikation (Folgerung, $\to$)

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

  ergänzen: Äquivalenz ($\leftrightarrow$), Antivalenz (xor)
- dreistellig: die Majorität (ergänze Wertetabelle)
- auf beliebig viele Argumente kann man erweitern: Disjunktion, Konjunktion, Äquivalenz, Antivalenz.
  Beispiel: $f(x_1, x_2, x_3, x_4) = f(f(f(x_1, x_2), x_3), x_4)$
- Majorität für ungerade viele Argumente

Rechenregeln

- Für $(\texttt{falsch}, \texttt{wahr}, \vee, \wedge)$ gelten viele der Rechenregeln wie für Zahlen $(0, 1, +, \cdot)$.
  Nennen Sie einige, überprüfen Sie diese (Wertetabelle!)
- ... und noch weitere, die für Zahlen nicht gelten, z.B.
  - Idempotenz der Disjunktion: $x \vee x = x$
  - das De-Morgansche Gesetz: $\neg x \wedge \neg y = \neg(x \vee y)$
    Augustus De Morgan (1806–1871),
    `http://www-history.mcs.st-andrews.ac.uk/Mathematicians/De_Morgan.html`

Boolesche Ausdrücke (Terme)

- De Morgan-Gesetz motiviert Unterscheidung zwischen
  - Semantik (Wert): Boolesche Funktion (Wertetabelle)
  - Syntax (Form): Darstellung der Funktion als Term
- für eine Menge V von Variablen definieren wir die Menge Term(V) der Booleschen Terme über V durch:
  - jeder Variable $v \in V$ ist in Term(V)
  - wenn f der Name einer k-stelligen Booleschen Funktion ist und $t_1, \ldots, t_k \in$ Term(V), dann ist $f(t_1, \ldots, t_k)$ auch in Term(V).
- jedes $t \in$ Term(V) ist ein *Baum*, Blätter (unten!) sind Variablen und Konstanten, innere Knoten sind beschriftet mit Operator-Namen, die Wurzel ist oben!

Auswertung von Termen

- Def: eine *Belegung* für die Variablenmenge V ist eine Funktion $b : V \to \mathbb{B}$.
  Bsp: $V = \{x, y\}$, $b = \{(x, 0), (y, 1)\}$.
- der *Wert* von $t \in \text{Term}(V)$ unter Belegung $b : V \to \mathbb{B}$ wird bezeichnet mit $\text{Wert}(t, b)$ und bestimmt durch
  – Einsetzen der Variablen-Werte (in den Blättern)
  – und Auswerten der Operatoren (von unten nach oben)
  Bsp: $\text{Wert}(\neg x \vee y, \{(x, 0), (y, 1)\})$
- jeder $t \in \text{Term}(V)$ mit $V = \{v_1, \ldots, v_k\}$ (Reihenfolge!) beschreibt eine k-stellige Boolesche Funktion $\mathbb{B}^k \to \mathbb{B}$.
  Bsp: Term $\neg x \vee y$ mit Reihf. $[x, y]$ beschreibt Implikation

Eigenschaften von Booleschen Termen

- ein Term $t \in \text{Term}(V)$ heißt allgemeingültig, falls für *jede* Belegung $b : V \to \mathbb{B}$ gilt: $\text{Wert}(t, b) = 1$. Bsp: $x \vee \neg x$
- . . . erfüllbar, falls es *eine* Belegung $b : V \to \mathbb{B}$ gibt mit: $\text{Wert}(t, b) = 1$ Bsp: $(x \vee y) \wedge (\neg x \vee \neg y)$
- . . . widersprüchlich, falls es *keine* Belegung $b : V \to \mathbb{B}$ gibt mit: $\text{Wert}(t, b) = 1$ Bsp: $x \wedge \neg x$
- Terme $t_1, t_2 \in \text{Term}(V)$ heißen *äquivalent* (wertverlaufsgleich), wenn für jede Belegung $b : V \to \mathbb{B}$ gilt: $\text{Wert}(t_1, b) = \text{Wert}(t_2, b)$
  Bsp: $x \wedge y$ ist äquivalent zu $y \wedge x$

Basis-Funktionen, Standard-Basis

- Def: eine Menge M von Operatore ist *Basis*, falls zu jedem Booleschen Term t ein äquivalenter t' nur mit Operatoren aus M existiert.
- Satz: Diese Funktionen bilden eine (Standard-) Basis:
  – Negation (Nicht) (einstellig)
  – Disjunktion (Oder) (beliebig viele Stellen)
  $\texttt{Oder}(x_1, \ldots, x_n) = 1$ gdw. wenigstens ein $x_i = 1$
  – Konjunktion (Und) (beliebig viele Stellen)
  $\texttt{Und}(x_1, \ldots, x_n) = 1$ gdw. alle $x_i = 1$
- Beispiele: $(x \to y) = \text{Oder}(\text{Nicht}(x), y) = \neg x \vee y$, $(x \leftrightarrow y) = (\neg x \wedge \neg y) \vee (x \wedge y)$

Standard-Basis (Beweis-Idee), Disjunktive NF

- Man stellt f als *Disjunktion* von Funktionen $f_1, f_2, \ldots$ dar (je eine Funktion für jede 1 im Werteverlauf von f)

x	y	$f(x,y)$	$f_1(x,y)$	$f_2(x,y)$
0	0	1	1	0
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

- Jedes f_i hat genau eine 1 im Werteverlauf, kann als *Konjunktion* von *Literalen* (Variable und negierten Var.) dargestellt werden. $f_1(x, y) = \neg x \wedge \neg y$, $f_2(x, y) = x \wedge y$ jede solche Konjunktion heißt *Klausel*
- insgesamt: $f(x, y) = (\neg x \wedge \neg y) \vee (x \wedge y)$ heißt *kanonische disjunktive Normalform* von f

Schaltungen

- (G/S 5.3) Boolesche *Schaltung* (Schaltnetz) mit k Eingängen ist Anordnung von *Schaltgliedern* (Gattern)
  – jedes Gatter hat Eingänge und einen Ausgang und ist mit einer Booleschen Funktion bezeichnet
  – jeder Gatter-Eingang ist verbunden mit: einem Schaltungs-Eingang oder einem *früheren* Gatter-Ausgang (kreisfrei!)
  – ein Gatter-Ausgang ist der Schaltungs-Ausgang
- Bsp: Eingänge x_0, x_1, Gatter (jeweils Ausgang zuerst): $x_2 = \neg x_0, x_3 = \neg x_1, x_4 = x_2 \wedge x_1, x_5 = x_0 \wedge x_3, \underline{x_6} = x_4 \vee x_5$
- Auswertung unter Belegung: ähnlich wie Term
- realisiert k-stellige Boolesche Fkt. (Bsp: Antivalenz, Xor)

Schaltung für binäre Addition

- Beispiel: Addition von Binärzahlen $(c_2, z_1, z_0) = (x_1, x_0)_2 + (y_1, y_0)_2$
  (3 Ausgänge links, 4 Eingänge rechts von „= “)
- realisiert durch Schaltung
  $z_0 = \text{Xor}(x_0, y_0),\ c_1 = x_0 \wedge y_0$
  $z_1 = \text{Xor}(x_1, y_1, c_1),\ c_2 = \text{Maj}(x_1, y_1, c_1)$
- dabei können Xor und Maj durch Gatter aus Nicht, Und, Oder ersetzt werden
  für Xor$(,)$: vorige Folie. für Xor$(,,)$ und Maj$(,,)$: Übung
- es genügt sogar *ein* Gatter-Typ! (nächste Folie)

Eine Basis mit einem Element

Die Funktion Nand : $(x, y) \mapsto \neg(x \wedge y)$ bildet eine Basis.

x	y	$x \wedge y$	$\text{Nand}(x,y)$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Beweis: wir können diese Funktionen darstellen:

- Nicht: $\neg x = \text{Nand}(x, x)$
- Und: $x \wedge y = \neg\,\text{Nand}(x, y) = \ldots$
- Oder: $x \vee y = \text{Nand}(\neg x, \neg y) = \ldots.$

. . . und nach vorigem Basis-Satz auch alle anderen Ftk.

nand in Hardware

- Nand mit drei Transistoren `https://www.allaboutcircuits.com/vol_4/chpt_3/5.html`.
- vier solche Nand in einen Schaltkreis (z. B. `https://www.ti.com/lit/ds/symlink/sn74f00.pdf`)
- . . . und daraus CPUs (benutzt Basis-Satz!)
  – aus Einzel-Elementen : mechanisch: 186? Charles Babbage: Analytical Engine, 1936 Konrad Zuse Z1; elektromechanisch: 1940 Z2 (Relais), elektronisch: 1943 ENIAC (John von Neumann) Elektronenröhre, 1955 Transistor
  – (1960) aus Standard-Schaltkreisen (nand, . . .)
  – (ab 1970) aus Spezial-Schaltkreisen

Aufgaben

WS21: Aufgaben 1 bis 3

1. für den Term $(\neg x \vee y) \wedge (z \to x)$:
   - den Syntaxbaum angeben,
   - den Wahrheitswert unter der Belegung $\{(x,0),(y,1),(z,1)\}$ angeben
   - die Wertertabelle der durch diesen Term beschriebenen dreistelligen aussagenlogischen Funktion angeben.
   - ist der Term allgemeingültig? erfüllbar? widersprüchlich?
   - einen äquivalenten Term in der Standard-Basis angeben.

2. Eine zweistellige Funktion f heißt *assoziativ*, wenn für alle x, y, z gilt: $f(f(x,y),z) = f(x,f(y,z))$.

   Stellen Sie mittels Wertetabellen fest, ob diese zweistelligen Funktionen assoziativ sind:

   a) Antivalenz, b) Implikation, c) Nand.

   Sind die Terme Maj(Maj(x_1,x_2,x_3), Maj(x_4,x_5,x_6), Maj(x_7,x_8,x_9)) und Maj$(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$ äquivalent?

   (Wieviele Zeilen hat die Wertetabelle? Man kann die Aufgabe lösen, ohne diese Tabelle komplett hinzuschreiben.)

3. Für Maj(x_1,x_2,x_3,x_4,x_5) an:
   - wieviele Klauseln enthält die disjunktive Normalform? (systematisch abzählen oder ausrechnen, Klauseln nicht alle einzeln aufschreiben)
   - einen (möglichst kleinen) äquivalenten Term in der Standard-Basis angeben
   - eine (möglichst kleine) äquivalente Boolesche Schaltung in der Standard-Basis angeben
   - verallgemeinern Sie die Schaltung auf Maj$(x_1,\ldots,x_7)$ und evtl. größere.

4. Begründen Sie, daß die Menge der Funktionen

   $\{$ Implikation (2-stellig), 0 (0-stellig) $\}$

   eine Basis bildet. Stellen Sie

   a) die 2-stellige Konjunktion, b) die 3-stellige Disjunktion

   in dieser Basis dar.

5. (Zusatz) Sieben Zwerge $1,2,\ldots,7$ bekommen im Schlaf je einen Hut auf den Kopf gesetzt, der blau oder rot ist.

   Die Zwerge wachen auf.

   Jeder Zwerg i sieht alle mit kleinerer Nummer als i. (Das bedeutet unter anderem: Zwerg 1 sieht niemanden. Zwerg 7 wird von niemandem gesehen. Niemand sieht seinen eigenen Hut.)

   Nun soll (in irgendeiner Reihenfolge) jeder Zwerg laut (d.h., hörbar für alle) eine Farbe (blau oder rot) nennen. (Wirklich nur dieses eine Bit Information, keine versteckten Nachrichten.)

   Die Zwerge sollen vor dem Schlafengehen einen Algorithmus vereinbaren, nach dem garantiert wenigstens 6 von ihnen die eigene Hutfarbe dabei richtig nennen.

   Hinweis: die mehrstellige Antivalenz-Funktion.

   Warum kann man 7 Richtige nicht garantieren?

Informationsdarstellung — Medien

Einleitung, Überblick

- Mensch kann unterschiedliche Signale erzeugen und wahrnehmen und Informationsgehalt zuordnen, z.B.
  - akustisch (Luftdruckschwankungen) — Mund → Ohren
  - optisch (Helligkeitsänderungen) — Gliedmaßen (Gestik), Gesicht (Mimik) → Augen
- mit Hilfsgeräten kann man diese Signale abspeichern (Mikrofon, Schellack-Platte; Schreibfeder, Schriftstück), vervielfältigen, wiedergeben (Grammophon, Telegraph)
- der Computer ist ein solches Hilfsgerät, Näherungen der Signale werden durch Bitfolgen repräsentiert.
- Gumm/Sommer 1.3 Inform., 1.5 Hardware, (11 Grafik)

Bausteine/Formen elektronischer Medien

- Zeichen: werden numeriert, die Nummer wird codiert, d.h., binär repräsentiert, Bsp: UTF-8, als Folge v. Bytes
- Text als Zeichenfolge
- (Baum-)strukturierter Text, Bsp: nroff (1972), LaTeX (1984), texinfo (1986), HTML (1993), Markdown (2004)
- Musikstücke: als Folge von Amplituden, z.B. pro Sekunde 48000 Zahlen mit 16 Bit (File-Format WAVE)
- Standbilder: rechteckige Anordnung von Farbpunkten (Pixeln); Farbe $= \mathbb{B}$ für Schwarz/Weiß, (Format BMP)
- Bewegtbilder: Folge von Standbildern
- Mischformen: Untertitel in Video, Audio in Webseite, ...
- Umwandlungen, z.B. `pdflatex`: strukt. Text → Grafik
- Kompression: ohne Verlust (zip,flac), mit (mp3,jpeg,h264)

Text-Strukturierung und -Formatierung

- Struktur (Kapitel, Abschnitte usw.) (schließlich) durch Formatierung, d.h., grafische Gestaltung ausdrücken.
- Bearbeitungsmodus WYSIWIG (what you see is what you get): man editiert direkt die grafische Form.
  Bsp: MS-Word, Libreoffice, Forum in Opal.
  - benötigt (simuliertes) Grafik-System zur Bedienung
  - viele Bedienvorgänge mit der Maus (zeitraubend)
  - man beschreibt die Formatierung, nicht die Struktur
- Textbeschreibungssprache, Bsp: LaTeX, Markdown.
  - man schreibt Quelltext (mit Tastatur, ohne Maus)
  - dieser Text beschreibt die logische Struktur
  - separate Übersetzung zu Grafik (Formatierung)
- Gumm/Sommer 1.7.3, 1.7.5

Schwarz-Weiß-Bilder

- eine Schwarz/Weiß-Rastergrafik ist eine Funktion $\Omega \to \mathbb{B}$ mit Definitionsbereich $\Omega = [0 \dots h-1] \times [0 \dots w-1]$
  (Rechteck mit Höhe $h \in \mathbb{N}$, Breite $w \in \mathbb{N}$)
  gleichwertig: ein Bild ist eine Folge von h Zeilen, eine Zeile ist eine Folge von w Bits.
- Beispiele: $h = w = 10$, Funktionen:
  $p(y,x) = (x \geq 5)$, $p(y,x) = (x \geq y)$,
  $p(y,x) = (x \geq 5 \vee y \geq 5)$, $p(y,x) = (x \equiv y \pmod 2)$

Beschreibung (Erzeugung) von Bildern

- Ziel ist oft Rastergrafik (für LCD/LED-Bildschirm, Laserdrucker, Schaltkreisbelichtung)
  aber nicht immer (Plotter: Stiftbewegungen)
- zur Bildkonstruktion sind andere Methoden nützlicher, die von einzelnen Pixeln abstrahieren
  - Vektorgrafik (Gumm/Sommer 11.2): Bild wird beschrieben durch Zeichenbefehle (für Linien, Kurven)
  - (diese VL jetzt) *algebraisch*: ... durch Termbäume mit
    - elementaren Bildern in Blättern
    - Operationen in inneren Knoten
- wir betrachten in dieser VL nur S/W-Bilder und einfache Operationen, da es noch andere VL zu Grafik geben wird.

Operationen auf Bildern

- elementare Bilder, z.B. Rechteck, Kreis
- aus bereits vorhandenen Bildern neue erzeugen durch
  - Neben-, Übereinandersetzen (row, column)
  - Transformation (ein Bild → ein Bild), z.B. Drehen
  - Kombination (mehrere Bilder → ein Bild), durch Boolesche Operationen auf Pixeln
- damit Bilder beschrieben durch Ausdrücke wie

```
Transform Rotate
  (Column [ Circle 100, Rectangle (100,100) True])
```

- zur Erinnerung: mathematische Kulturtechnik: unterscheide Syntax (Term) von Semantik (hier: Bild)

Neben- und Übereinandersetzen

- für Bilder $p_1, \dots, p_n$ bezeichnet row$[p_1, \dots, p_n]$ das Bild, das durch Nebeneinandersetzen der p_i entsteht
- die Höhen der p_i müssen dabei übereinstimmen, damit ein Rechteck entsteht.
- $p =$ row$[p_1, p_2]$ mit $p(y,x) =$ wenn $x <$ width(p_1), dann $p_1(y,x)$, sonst $p_2(y, x -$ width$(p_1))$
- Übung: col$[p_1, \dots, p_n]$ Übereinandersetzen analog
- Satz: Das zweistellige col ist: assoziativ.
- Das zweistellige col ist nicht kommutativ. (Beispiel?)
- Test: row$[$col$[p,q]$, col$[r,s]] =$ col$[$row$[p,r]$, row$[q,s]]$?

Boolesche Kombination von Bildern

- wenn f eine k-stellige Boolesche Funktion ist,
  dann bezeichnet `combine f`$[p_1, \dots, p_n]$
  das Bild p mit $p(y,x) = f[p_1(y,x), \dots, p_n(y,x)]$.
- alle p_i müssen übereinstimmende Abmessungen haben, damit diese Operation wohldefiniert ist.
- für Boolesche Operationen auf Bildern gelten die gleichen Gesetze wie für Boolesche Operationen auf Wahrheitswerten,
- Ü: z.B. das Gesetz von De Morgan.

Transformationen von Bildern

- Spiegeln: $q = \mathsf{Transform}\,\mathsf{Mirror}\,p$
  mit $q(y,x) = p(y, \mathsf{width}(p) - 1 - x)$
- Skalieren: $q = \mathsf{Transform}(\mathsf{Scale}(f,g))p$
  mit $q(y,x) = p(\lfloor y/f \rfloor, \lfloor x/g \rfloor)$
- Drehen: $q = \mathsf{Transform}\,\mathsf{Rotate}\,p$
  mit $q(y,x) = p(x, \mathsf{height}(p) - 1 - y)$
- Teilbild ausschneiden: selbst ausprobieren, Beispiel

```
Transform (Slice{size=(200,200),start=(0,0)}) (Circ
```


Hausaufgaben

1. (nochmals) die Aufgabe „wieviele Bit pro Semester“: vergleichen Sie unterschiedliche Medienformen:
   - (Wdhlg.) jede Folie mit 1 Byte pro Zeichen, 10 Zeilen, 50 Spalten
   - jede Folie einzeln als Schwarz-Weiß-Bild mit 300 dpi (dots per inch) auf A4 (quer)
   - jede Vorlesung als Audio-Stream (mono, 48 kHz Sample-Rate, 16 Bit je Sample)
   - jede Vorlesung als Video-Stream (HD-Format, für jedes Pixel 4 Byte Farb-Information, 30 fps (frames per second))

   Nur jeweils die Anzahl der Bit für eine Vorlesung (20 Folien bzw. 90 min) ausrechnen und die Verhälnisse

angeben, auf Zehnerpotenzen gerundet.
Die dabei entstehenden Zahlen sind unrealistisch groß. In der Praxis werden solche Daten (teilweise drastisch) komprimiert, das behandeln wir später.

2. Benutzen Sie die algebraische Bild-Sprache (die Test-Aufgabe im autotool) zum Zeichnen von
   - einzelnen Buchstaben
   - Verkehrszeichen

   Verwenden Sie dabei auch Kreise, Bild-Ausschnitte (slice), sowie Boolesche Verknüpfungen.
   Laden Sie einige so erzeugte Bilder im Forum des Opal-Kurses hoch, aber *ohne* Quelltexte. Diese sollen von den anderen Studenten geraten werden. Sie können Hinweise geben (z.B., welche Operationen wurde wie oft

benutzt). Auflösung dann in der Übung.

3. Überprüfen Sie das De-Morgansche Gesetz (aus der Aussagenlogik) durch die entsprechende Konstruktion auf Bildern.
   Formulieren Sie Assoziativ- und Distributiv-Aussagen für Bild-Operationen (Row, Column, Boolesche Kombinationen) und überprüfen Sie diese an Beispielen.

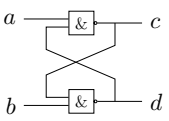
Speicher

Motivation, Überblick

- Ziel: (Repräsentation von) Information bereithalten für spätere Verarbeitung (Nanosekunden bis Jahrzehnte)
- Realisierungen (Beispiele):
  - Register der CPU für Maschinenzahlen
  - Hauptspeicher (RAM)
  - interne, externe Laufwerke
- Kennzahlen (Bsp: eine aktuelle interne SSD, SATA)
  - Zugriffszeit (einzeln, im Block) (Bsp: 500 MB/s)
  - Kapazität (Bsp: 1 TB)
  - Kosten (Einkauf/Betrieb) (Bsp: 110 EUR / $\approx$ 0 EUR)

Nand-Flip-Flop als 1-Bit-Speicher

- Schaltwerk (sequentielle Schaltung): mit Rückführungen (bisher betrachtet: Schaltkreis, Schaltnetz: ohne Rückf.)
- Tabelle rechts: Ausgänge (c, d) nach Eingangsbelegungen (a, b) in angegebener Reihenfolge



a	b	c	d
0	1	1	0
1	1	...	...
1	0	...	...
1	1	...	...

- bei Eingang $a = 1, b = 1$ zeigt der Ausgang c, ob die letzte 0 von a oder b kam

  d.h., dieses Schaltwerk speichert 1 Bit

Physikalische Grundlagen für Speicher

- Speicher wird realisiert durch physikalisches System, dessen Zustand geändert werden kann. — Derzeit:
- Halbleiter-Speicher
  - statisch (Flip-Flop), Zustand: Potential
  - dynamisch (RAM, SSD: Transistor als Kondensator), Z: Ladung
  - irreversibel (ROM), Z: vorhandene/unterbrochene elektr. Verbindung
- magnetische Speicher (Magnetband, Diskette, Festplatte)

  Zustand: Orientierung eines magnetischen Dipols
- optische Speicher (CD, DVD, BD), Zustand: mechanisch (Erhebung/Vertiefung, ändert Reflektion)

Speicherprinzipien historischer Rechner

- mechanische Speicher:

  (Zahnräder, Hebel: Pascal 1650, Babbage 1823, Zuse (Z1) 193?)

  (Lochkarte: Jaquard 1805; Lochband)
- elektromechanisch (Relais), Zuse (Z2/Z3) 1936;

  elektronisch (Röhren)
- magnetisch:

  fest: Ferritkerne,

  rotierend: Magnettrommeln

Disketten und Festplatten

- Speichermedium: magnetisch beschichtete Scheibe(n)
- Speicherblock adressiert durch
  - Nummer des Kopfes (Scheibe, Seite (oben/unten))
  - Nummer der Spur (des Zylinders) (Kopfposition)
  - Nummer des Sektors
- Hausaufgabe: für eine handelsübliche HDD (3.5 Zoll, 8 TB, 4 doppelseitige Scheiben, 8 Köpfe, 7200 U/min)
  - maximale Zugriffszeit (ohne Cache)? Wieviele Befehle kann die CPU derzeit ausführen?
  - Mit welcher Geschwindigkeit bewegt sich die Magnetschicht relativ zum Kopf?
  - Wieviel Fläche pro Bit? (als Quadrat – Seitenlänge?)

CD und DVD

- *Prinzip*: entlang einer Spur gibt es Vertiefungen (pits) und Erhebungen (lands), bei Abtasten wird ein Laserstrahl reflektiert (land) oder nicht (pit).
- Vertiefungen sind „oben“, darauf ist Schutzschicht (Lack), abgetastet wird von unten.
- CD: pit-Länge: 0.8 μm, Spur-Abstand: 1.6 μm
- wie lang ist die Spirale? wie schnell dreht sich die CD? wieviel Bit (pits) passen drauf?
- DVD: pits und Abstand $\approx$ halb so groß wie bei CD
- DVD-9 verwendet zwei Schichten (Laser muß anders fokussiert werden, erste Schicht: innen nach außen, zweite: außen nach innen).

Die Speicher-Hierarchie

- Ein Computer benutzt Speicher dieser Art:

  1. CPU-Register
  2. Hauptspeicher
  3. Festplatte (intern)
  4. Archiv-Speicher (extern)
- dabei gilt (von oben nach unten):
  - Zugriffszeit steigt,
  - Größe steigt,
  - Kosten (pro Bit) sinken.

Cache

- Idee: langsamen (umfangreichen, preiswerten) Speicher S beschleunigen durch (wenig) zusätzlichen, schnelleren (teureren) Speicher C
- – C enthält Kopien von einigen Bereichen von S
  - bei jedem Zugriff auf S wird stattdessen die (schnellere) Kopie in C benutzt — falls sie vorhanden ist.
  - ... falls nicht, wird ein länger nicht benutzter C-Bereich aufgegeben und mit dem jetzt aktuellen Bereich (= Daten aus der Nähe des aktuellen Zugriffs) gefüllt.
- obiges gilt für Lese-Zugriffe. Geschrieben wird meist auf C *und* S (write-through)

Speicher-Zugriffs-Arten (technisch)

- (digitaler) Speicher s enthält Folge von Bits,
  aber die Kosten einzelner Operationen in einer Zugriffsfolge $s[i_0], s[i_1], s[i_2], \ldots$ können je nach technischer Realisierung sehr unterschiedlich sein
- – wahlfreier Zugriff (random access memory, RAM)
  alle Zugriffe gleich schnell
  – sequentieller Zugriff (Magnetband, Lochband)
  $s[a], s[a+1], s[a+2], \ldots$ (aufeinanderfolgende Indizes)
  erster langsam, weitere schnell
  – index-sequentieller Zugriff (wahlfreier Zugriff auf Blöcke mit sequentiellem Inhalt) (Festplatten)
  – RAM mit Cache: ähnlich zu index-sequentiell

Speicher-Zugriff (Anwender-Sicht)

- die auf Festplatte gespeicherte Information ist ...
  – (Implementierungs-Sicht) . . . eine Bitfolge.
  – (Anwender-S.) . . . ein (Verzeichnis-)Baum von Dateien
- eine Datei . . .
  – (Implementierungs-Sicht) . . . ist eine Bitfolge
  – (Anwender-Sicht) . . . repräsentiert Information, deren Art (Text, Grafik, Musik, . . .) erkennbar ist an:
  Endung des Dateinamens (Bsp: `jpeg`),
  Beginn der Bitfolge, Bsp:

```
377 330 377 340  \0 020   J   F   I   F  \0 001 00
d8ff    e0ff     1000    464a    4649    0100    0
```


Dateisysteme

- *Betriebssystem* ist Schnittstelle zwischen Hardware und Benutzer bzw. Benutzerprogrammen (später mehr dazu)
- *Dateisystem:* Schnittstelle zu nichtflüchtigem Speicher
- technische Realisierung durch Speicherblöcke
  – Datei: Info-Blöcke, Inhalts-Blöcke
  Info-Block enthält Verweise auf Inhalts-Blöcke
  – Verzeichnis ist auch Datei, Inhalt ist
  Abbildung von Name auf (ersten) Info-Block
  – Verwaltung freier (nicht (mehr) benutzter) Blöcke
- Beispiel: eine Datei „löschen“: Verzeichnis-Eintrag entfernen, Blöcke als frei markieren.— Daten sind rekonstruierbar! (bis Blöcke neu beschrieben werden)

Dateistruktur und Suche

- Navigation im Verzeichnisbaum: setzt Struktur voraus (was steht wo), an die man sich auch halten muß
- bequemer scheint: strukturlose Datei-Ablage,
  dann Suche nach Dateinamen, Metadaten, Inhalt
- wenn gesuchtes nicht auf eigenem PC gefunden (oder schon vorher), dann „im Internet“ (weiter)suchen
- . . . um den Benutzer auf der eigenen Suchmaschine überwachungswirtschaftlich zu verwerten
  vgl. Aussagen zu Cortana in
  `https://www.eff.org/deeplinks/2016/08/windows-10-microsoft-blatantly-disregards-u`

Speicher und Speicherdienste

- Desktop-PCs: standardisierte Schnittstellen,
  z.B. SATA für SSD/HDD, USB für externe Speicher
  Wartung und Erweiterung d. Austausch v. Komponenten,
  Preisfindung und Innovation durch Marktkräfte
- das wollen die Hersteller natürlich vermeiden, z.B. durch
  – firmenspezifische Schnittstellen,
  – fest verbaute/verlötete Komponenten
  – Garantieverlust bei Eigenbau
- Bsp: Tablets (E-Ink-Reader) mit extra wenig Speicher
  . . . um den Käufer auf Cloud-Speicherdienst des Herstellers umzuleiten . . . um ihn dabei überwachungswirtschaftlich zu verwerten

Hausaufgaben

1. Aufgaben von Folie „Disketten und Festplatten“
2. Welchen Informationgehalt (in Bit) hat ein Buch? Den Informationsgehalt einer Buchseite abschätzen als Bild (300 dpi schwarz/weiß).
   Welche Informationsdichte (in Bit pro cm^3)?
   Vergleichen mit Informationsdichte einer HDD (mit Gehäuse), eines Paketes mit SD-Karten (vgl. Randall Munroe, *What If?*, `https://what-if.xkcd.com/31/` – dort angegebene Zahlen nachrechnen)
3. Wiederholung Schaltkreise: lösen Sie autotool 45-1 (Majorität von 5 Eingängen) durch möglichst kleine Schaltung.

Ansatz: zu realisieren ist die Funktion $\mathsf{Atleast}_3^5(x_0, \ldots, x_4)$ mit $\mathsf{Atleast}_k^n(x_0, \ldots, x_{n-1}) :=$ wenigstens k der n Eingaben sind wahr.

Ergänzen, erklären und verallgemeinern Sie:

- $\mathsf{Atleast}_2^3(x_0, x_1, x_2) =$
  $(\mathsf{Atleast}_1^2(x_0, x_1) \wedge x_2) \vee (\mathsf{Atleast}_2^2(x_0, x_1) \wedge \neg x_2)$.
  Hier kann man „$\wedge\neg x_2$“ sogar weglassen.
- $\mathsf{Atleast}_1^2(x_0, x_1) = \ldots$
- $\mathsf{Atleast}_2^2(x_0, x_1) = \ldots$

Die Knoten der gesuchten Schaltung sind $\mathsf{Atleast}_k^n(x_0, \ldots)$ mit $1 \le k \le 3, 2 \le n \le 5, k \le n$.

Programme

Motivation

- (weiterhin) semantischer Bereich: rechteckige Schwarz-Weiß-Bilder; semantische Operationen: neben, über, ...; Syntax: algebraische Ausdrücke (Term-Bäume)
- ist unpraktische Notation, Beispiele:
  - Ziffer 8 als zwei Kreisringe: doppelter Teil-Term
  - Schachbrett: 8 Zeilen, jede Zeile 8 Quadrate
- *Programm* mit Namen als Bezeichner von Werten

```
let { ring = ... } in Column [ring, ring]
```

- Programm-Ausführung auf Computer ersetzt „mechanische“ (repetitive) Arbeit des Menschen

Geradeaus-Programme

- verbal: a bezeichnet einen Kreis, b bezeichnet zwei a nebeneinander, setze drei b übereinander. — formal:

```
let { a = Circle 100 ; b = Row [a,a] }
in  Column [b,b,b]
```

- Syntax: ein Let-Ausdruck besteht aus
  - Folge von Definitionen (nach `let`, in `{ }`)
  - Körper (nach `in`)
- Syntax: eine Definition hat die Form *Name* = Ausdruck
  Semantik: ab der nächsten Definition (und im Körper): der definierte Name bezeichnet den Ausdruck
- vgl. die Notation von Schaltkreisen

Sichtbarkeit, statische Korrektheit

- ein Ausdruck ist *statisch korrekt*,
  wenn jede Benutzung eines Namens
  im Sichtbarkeitsbereich einer gleichnamigen Definition ist
- nicht statisch korrekt sind:

```
Row [Circle 100, a]
let {a = Circle 100} in b
let {b = Row [a] ; a = Circle 100} in b
```

- ein Ausdruck ist *dynamisch korrekt*, wenn bei Auswertung keine Fehler der Domain-Semantik auftreten
  (z.B. `Column` mit verschieden breiten Argumenten)

Beispiel: Schachbrett

- ohne Namen: langer und langweiliger Programmtext

```
Column [ Row [ Rectangle (100,100) False
             , Rectangle (100,100) True, ... ]
       , ... ]
```

- mit Namen:

```
let { s = Rectangle (100,100) False
    ; w = Rectangle (100,100) True
    } in Column [Row [s,w,s,...], Row [w,s,...], ..
```

- weitere Kompression möglich, z.B.

```
let { s = Rectangle (100,100) False
    ; w = Combine Not [s]
    ; r1 = Row [s,w,s,...]; r2 = Row [w,s,...]
    } in Column [ r1, r2, r1, ... ]
```


Baum-Struktur

- Programmtext = Ausdruck, jeder Ausdruck ist
  1. ein Domain-Ausdruck (Bsp: `Circle 100`, `Row [a,a]`)
     Domain-Operation, evtl. mit Argument-Ausdrücken
  2. oder ein Let-Ausdruck: nach = und Körper: Ausdrücke
  3. oder ein Name (Bsp: `a`)
     durch `let` definierte Namen (Bsp: `a`): Kleinbuchstaben, Operationsnamen (Bsp: `Row`): beginnen groß
- Let im Argument-Ausdruck:

```
Row [Circle 100, let {a=Circle 100} in a]
```

- Let in Def.: `let {a=let {b=Circle 100} in b} in a`
- Let im Let-Körper (und Abkürzung dafür):

```
let {a=Circle 100} in let {b=Row[a,a]} in b
let {a=Circle 100;         b=Row[a,a]} in b
```


Verdeckung von Namen

- Let-Blöcke können geschachtelt werden

```
let { a = Circle 100 } -- Definition 1
in  let { b = Row [a,a] } -- Definition 2
    in  let { a = Circle 200 } -- Definition 3
        in  Column [a,a] -- Bezug auf 1 oder 3?
```

- die jeweils innerste Definition eines Names (Bsp: 3) wird benutzt (ist *sichtbar* und *verdeckt* die äußeren (Bsp: 1)

- ```
  let { a = Circle 100 ; a = Circle 200 } in a
  let { a = Circle 100 ; a = Row [a,a] }  in a
  let { a = Circle 100
      ; b = let { a = Circle 200 } in Row [a,a]
      } in a
  ```


Unterprogramme (Motivation)

- Namen bezeichnen Werte, zur Wiederverwendung

```
let { k = Circle 100 } in Row [k,k,k,k]
```

- wie benennt man die Idee „viermal nebeneinander“?

```
let { k = Circle 100; l = Combine Not [k] }
in Column [ Row [k,k,k,k], Row [l,l,l,l] ]
```

- wir führen Parameter (Bsp: `p`) für Namen (Bsp: `vier`) ein:

```
let { k = Circle 100; l = Combine Not [k]
    ; vier p = Row [p,p,p,p] }
in Column [ vier k, vier l ]
```

- dieses `vier` ist ein *Unterprogramm*,
  es realisiert eine *Funktion* von Bild nach Bild

Unterprogramme (Syntax und Semantik)

- *Definition* eines Unterprogramms (UP):

```
let { ...; f p_1 ... p_n = b; ... } in _
```

  - definiert ein n-stelliges Unterprogramm mit Namen `f`.
  - Die Namen `p_i` heißen *formale Parameter*.
  - Der Ausdruck `b` heißt der *Körper* (die Implementierung).
  - die `p_i` sind nur in `b` sichtbar
- *Aufruf* eines Unterprogramms: der Ausdruck
  `f a_1 ... a_k` (mit Argument-Ausdrücken `a_i`)
  - ist *statisch korrekt*, wenn f Name eines UP und `k = n`
  - und hat den Wert von `let {p1 = a1; ...} in b` bei Auswertung von `b` bezeichnet formaler Parameter `p_i` den Wert des Argumente-Ausdrucks `a_i`
```

Beispiele für Unterprogramme

- geschachtelter Aufruf:

```
let { f x = Row [x, Combine Not [x], x] }
in  f (f (Circle 100))
```

- geschachtelte Definition (Def. von g benutzt f)

```
let { f x = Row [x,x,x,x]
    , g x = let { y = f x } in Column [y,y,y,y] }
in  g (Circle 100)
```

- zweistelliges Unterprogramm:

```
let { f x y = Column [Row [x,y], Row [y,x]] }
in  f (Circle 100) (Rectangle (100,100) True)
```


Warum nicht Java, C#, . . . ?

- Sie studieren nicht Informatik.
  auch für Informatiker sind das ungeeignete Start-Sprachen.
- Sie können nicht in Computer-Pool üben, wo wir Software installiert haben,
  wir wollen Ihnen eigene Installation nicht zumuten,
  und Sie auch nicht der überwachungswirtschaftlichen Ausbeutung durch Online-Programmier-Angeboten von Dritten aussetzen
- Sie üben online mit autotool,
  Daten bleiben an der Hochschule,
  Aufgaben passen genau zur Vorlesung.

Trotzdem: Vergleich und Quellen

- unsere Sprache für Bilder: `https://www.imn.htwk-leipzig.de/~waldmann/talk/21/abp`
  vgl. Paul Hudak: *The Haskell School of Expression*, 2000,
  `https://www.cs.yale.edu/homes/hudak/SOE/index.htm`
- Namen, Blöcke, Sichtbarkeiten: genau so in jeder anderen Sprache, z.B. Java,
  vgl. Gumm/Sommer 2.5 (Deklaration, Initialisierung, Kontexte, Audrücke, Auswertung)
  Dort zusätzlich: Typisierung (bei uns nächste Woche) und imperative Programmierung (machen wir nicht)
- Unterprogramme: genau so.
  vgl. Gumm/Sommer 2.5.6 (Funktionen). Dort zusätzlich: Verzweigungen, Rekursion (bei uns nächste Woche)

Hausaufgaben

- Benutzen Sie für alle Aufgaben die dazu passende oder ähnliche autotool-Aufgabe. Das autotool muß nicht „ja“ sagen, sondern nur Ihre Bilder zeichnen.

1. Ziffern, Buchstaben, Symbole (z.B. Verkehrszeichen) eigener Wahl programmieren

   unter wesentlicher Benutzung von Namen und Unterprogrammen.

2. möglichst große Bilder konstruieren (es geht hier nur um die Fläche, nicht die Pixelfarben)

   durch (kleine) Programme der Form

   ```
   let { pixel = Rectangle (1,1) True
   ```



```
    ; row x y = Row    [x,y]
    ; col x y = Column [x,y]
} in  a
```

wobei der Ausdruck `a` *keine* Domain-Operation enthält (sondern nur Let-Blöcke, UP-Aufrufe, Namen)

Betrachten Sie dabei diese syntaktischen Einschränkungen für `a`:

- überhaupt kein Let
- mit Let, aber alle dort definierten Namen 0-stellig (also keine Unterprogramme)
- . . . mit maximal 1-stelligen UP
- . . . mit maximal 2-stelligen UP

Beispiele 6, 7 aus `https://www.imn.`

`htwk-leipzig.de/~waldmann/talk/21/abp/`
vorführen, erklären, die Größe nachrechnen.

Falls Sie die eingestellte Größenbeschränkung im autotool überschreiten: rechnen Sie selbst (exakt) aus, welche Fläche Ihr Bild hat.

3. möglichst kleine Programme, die ein Schachbrett erzeugen. Diskutieren Sie die Verwendung von
   - Rechtecken und Antivalenz (`Combine Xor [p,q]`)
   - Let, Unterprogrammen

   Für 8×8, für größere Bretter.

Programm-Ablauf-Steuerung

Motivation

- bisher haben wir diese Bausteine für Programme:
  - Domain-Operationen, Bsp: Bilder: Row, Logik: And
  - Definition und Benutzung von Namen für Domain-Elemente und Domain-Funktionen
- wir können Domain-Operationen kombinieren (Resultat der einen ist Eingabe für die andere) durch
  - zusammengesetzte Terme (Termbäume)
  - Definitions-Ketten, Bsp: `{a = _; b = _ a _}`
  - Aufruf-Ketten, Bsp: `f (g x) (h (h y))`
- *daten-abhängige* Steuerung des Programmablaufs?

Beispiele für Daten-Abhängigkeiten

- abhängig von einem Wahrheitswert:

```
let { f x = if x
            then Rectangle (100,100) False
            else Circle 100
} in Row [ f True, f False, f True ]
```

- abhängig von einer natürlichen Zahl (im Bsp: die 4)

```
let { g x = Row [ x, Combine Not [x] ] }
in  iterate g 4 (Circle 100)
```

- Anwendung: Wahrheitswert/Zahlenwert kommt aus Benutzereingabe, d.h., ist zum Zeitpunkt der Programmierung nicht bekannt (vgl. Def. Algorithmus)

Datentypen und statische Typisierung

- vorige Beispiele motivieren Unterscheidung zwischen:

  Wahrheitswerten, Zahlen, Domain-Elementen (Bildern)

  bereits beim Programmieren: für jeden Namen und für jeden Teil-Ausdruck wird bei der *statischen Analyse* der *Typ* bestimmt—oder das Programm abgelehnt
- Programme wie `Column[Circle 100, False]`, die gar kein Bild bezeichnen, sondern einen Laufzeitfehler, werden dadurch bereits vor der Ausführung als *statisch fehlerhaft* erkannt (und können repariert werden)
- bei jedem technischen System: je später ein Fehler erkannt wird, desto teurer ist die Reparatur

Typ-Inferenz und Typ-Deklaration

- in Programmiersprache mit statischer Typisierung

  (Algol 1960, ML 1973, Haskell 1990, Java 1995, . . .)

  haben jeder Name und jeder Ausdruck einen Typ, wird
  - durch Computerprogramm (Compiler) bestimmt (type inference, ML, Haskell)
  - oder durch Programmierer notiert (deklariert) und durch Compiler geprüft (type checking, Algol, Java)
- für die Sprache in autotool-Aufgaben: Typen aller formalen Parameter deklarieren

```
let { a = Circle 100 ; b = 8
    ; f (x :: Pic) (y :: Bool) = Row [x, x] } in _
```


Die Verzweigung

- konkrete Syntax: `if` B `then` J `else` N
  dabei B (Diskriminante), J (Ja-Zweig), N (Nein-Zweig) beliebige Ausdrücke (Teilprogramme)
- abstrakte Syntax: ein Knoten mit Kindern B, J, N
- statische Semantik (Typisierung): für jedem Typ T gilt:
  - wenn $B :: \mathbb{B}$ (der Typ von B ist $\mathbb{B}$ = Wahrheitswerte) und $J :: T$ und $N :: T$ (Typ von J ist T, Typ von N ist T)
  - dann (`if` B `then` J `else` N) $:: T$
- dynamische Semantik (Auswertung):
  - wenn B den Wert 1 (True) hat, Notation: Wert$(B) = 1$
  - dann Wert(`if` B `then` J `else` N) = Wert(J)
  - sonst . . . = Wert(N).

Die Wiederholung (Iteration)

- konkrete Syntax: `iterate` F K A
  dabei F ein Name, K und A Ausdrücke
- abstrakte Syntax: Knoten mit drei Kindern
- statische Semantik (Typisierung): für jeden Typ T:
  - wenn $F :: T \to T$ (Typ von F ist Funktion von T nach T) und $K :: \mathbb{N}$ (Typ von K ist Zahl) und $A :: T$
  - dann (`iterate` F K A) $:: T$
- dynamische Semantik (Auswertung):
  - wenn Wert$(F) = f$, Wert$(K) = k$, Wert$(A) = a$,
  - dann Wert(`iterate` F K A) $= f^k(a)$

  wobei $f^0(a) = a$ und $\forall n \geq 0 : f^{n+1}(a) = f(f^n(a))$

Beispiele zur Iteration

- aus der Nachfolgerfunktion `succ :: Nat -> Nat`, Bsp. `succ 7 = 8`

  kann man viele arithmetische Fkt. durch Iteration erhalten
- die Addition: `f x y = iterate succ x y`
- die Multiplikation (wie ist die Lücke zu ersetzen?)

```
g x y = let {h z = f y z} in iterate h x _
```

- Ü: das Potenzieren ähnlich zum Multiplizieren
- und so weiter, d.h., mit kleinen Programmtexten *sehr* große Zahlen (und auch Bilder)

Termination

- in unserer Sprache *terminiert* jedes Programm P:
  die Auswertung von P liefert nach endlich vielen Schritten ein Resultat, keine Auswertung dauert unendliche lange.
- mehrfache Auswertung eines Teilprogramms nur durch Unterprogramm-Aufrufe und Iteration,
  - keine Rekursion: kein Unteprogramm kann sich selbst aufrufen, `let {f x = Column[f x,f x]} in _`
  - Anzahl der Iterationen steht vor Beginn der Iteration fest
- viele Programmiersprachen gestatten beliebige bedingungsgesteuerte Wiederholung, aber ich möchte die Sache hier einfach halten (so einfach wie möglich, aber nicht einfacher)

Iteration zur Bild-Erzeugung

- mehrmalige Anwendung einer Funktion f : Pic → Pic

```
let { f (x :: Pic)
      = Column [Row[x,x],Row[x,Combine Xor[x,x]]]
} in iterate 4 f (Circle 10)
```

- viele weitere Beispiele, evtl. Autotool-Aufgaben (Programmtext ergänzen)
- vgl. auch Gumm/Sommer 11.3.2 Turtle-Grafik, 11.3.3 L-Systeme,
  - dort: Rekursion entspricht bei uns: Iteration
  - dort: imperative Grafik (Stift wird bewegt), bei uns: deklarativ (Bilder werden kombiniert).

Invarianten (Definition, Motivation)

- Def: eine *Invariante* P einer Funktion $f : T \to T$ ist eine Eigenschaft (d.h., eine Funktion $P : T \to \mathbb{B}$), die bei Anwendung von f erhalten bleibt, d.h., für alle $x \in T$ gilt: wenn $P(x)$, dann $P(f(x))$
- Bsp: für $f(x) = x + 2$ ist die Eigenschaft $P(x) =$ *die Zahl* x *ist ungerade* invariant. Bsp: $P(3)$ ist wahr, $P(f(3)) = P(5)$ ist wahr.
- Satz: Wenn P eine Invariante von f ist, und $P(a)$ gilt, dann gilt für jedes $k \geq 0$, daß $P(\texttt{iterate}(f, k, a))$
- Anwendung: mit Invarianten beweist man Aussagen über Iterationen, die für jede Iterationszahl gelten.
- das Beweisprinzip ist vollständige Induktion, $P(a)$ ist Induktionsanfang, Invarianz im Induktionsschritt benutzt

Geometrische Invarianten (Beispiele)

- für die Funktion

```
f (x :: Pic) =
  Column [Row [Combine Not [x],x], Row[x,x]]
```

  sind diese Eigenschaften invariant:
  - $P(x) = x$ ist quadratisch
  - $Q(x) = x$ enthält gleichviele schwarze wie weiße Pixel
- beachte: über $Q($`iterate f 8 (Rectangle (1,1) False)`$)$ kann man damit nichts sagen, denn $Q($`Rectangle (1,1) False`$)$ gilt nicht.

Invariante (Beispiel)

- Behälter mit 20 weißen und 21 schwarzen Kugeln.
  Man zieht zwei Kugeln. Wenn diese gleichfarbig sind, entfernt man beide. Wenn diese verschiedenfarbig sind, legt man die schwarze Kugel zurück.
  Ist es möglich, daß durch eine Folge solcher Schritte der Behälter leer wird?
- mathematische Modellierung:
  Zustand des Behälters ist Zahlenpaar $(w, s) \in \mathbb{N} \times \mathbb{N}$.
  die möglichen Zustandsänderungen sind:
  - entferne zwei weiße: $G_w : (w, s) \to (w - 2, s)$
  - entferne zwei schwarze: $G_s : (w, s) \to (w, s - 2)$
  - eine schwarze zurück: $U : (w, s) \to (w - 1, s)$

Invariante (Beispiel—Lösung)

- Behälter mit 20 weißen und 21 schwarzen Kugeln.
  Man zieht . . . Ist es möglich, daß durch eine Folge solcher Schritte der Behälter leer wird?
- Zustand ist Zahlenpaar $(w, s) \in \mathbb{N} \times \mathbb{N}$.
  Startzustand ist $(20, 21)$
  - entferne zwei weiße: $G_w : (w, s) \to (w - 2, s)$
  - entferne zwei schwarze: $G_s : (w, s) \to (w, s - 2)$
  - eine schwarze zurück: $U : (w, s) \to (w - 1, s)$
- die passende Invariante ist: s ist ungerade.
  - gilt für den Startzustand $(20, 21)$
  - bleibt bei jeder Operation G_w, G_s, U erhalten
  - gilt *nicht* für den Zielzustand $(0, 0) \Rightarrow$ ist unnerreichbar

Invarianten (Hausaufgabe)

- In einer Stadt leben 10 Anhänger der Partei A, 11 der Partei B, 12 der Partei C.
  Jedesmal, wenn sich zwei Anhänger unterschiedlicher Parteien treffen, diskutieren diese so lange, bis beide zu Anhängern der dritten Partei werden.
  Ist es möglich, daß nach einer Folge solcher Treffen alle Parteien gleich viele Anhänger haben?
- Hinweise: wenn ja, dann geben Sie eine solche Folge an. Wenn nein, dann gehen Sie eine Eigenschaft $P : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ an, die für jedes einzelne Treffen invariant ist und anfangs gilt, aber schließlich nicht. Benutzen Sie Teilbarkeit durch 3.

Vergleich, Quellen

- Gumm/Sommer: 2.5.7 Typfehler, 2.6.2 Kontrollstrukturen (ist leider die falsche Übersetzung von *to control*, richtig ist *control flow* = Programm- Ablauf- *Steuerung*)
- dort imperative Programmierung, bei unseren Programmen gibt es keine Speicherzustandsänderungen (Nebenwirkungen, falsche Übersetzung: Seiteneffekte)
- im Informatik-Studium programmiert man gelegentlich imperativ, um zu verstehen, wie die heutige Hardware funktioniert, aber Ziel der Anwendungsprogrammierung ist gerade die Abstraktion von der Hardware
- zur Abschreckung auch Sprachen ohne statisches Typsystem (d.h., unsicher, ineffizient): Javascript, Python

Hausaufgaben

1. Arithmetische Funktionen durch Iteration aus der Nachfolgerfunktion: Addieren, Multiplizieren, Potenzieren und weitere arithmetische Funktionen implementieren und vorführen.
   Mit einem kleinen Programm, das nur `iterate`, `succ` und `let` verwendet, eine möglichst große Zahl erzeugen.
2. durch Iterieren von Bildfunktionen (unter Verwendung von Row, Column, Mirror, Rotate, Xor; nicht unbedingt alle gleichzeitig) interessante Muster erzeugen.
   Beispiel

```
let { f (x::Pic) =
```

```
            Column [Row  [Transform Mirror x
                   ,Row [Transform Rotate x,
 } in  iterate f 4 (Triangle (20,20) False)
```

Bilder veröffentlichen (vor der Übung), Programme nicht oder nur Hinweise.

3. Die Aufgabe mit Parteien A,B,C.

Algorithmen

Motivation, Überblick

- am Beispiel Suchen und Sortieren: Einblick in Fragen
  - Korrektheit, Ressourcenverbrauch

  und Methoden
  - Spezifikation, Rekursion/Induktion

  der Synthese und Analyse von Algorithmen
- wir abstrahieren von konkreter Programmierung und von konkreten Datenstrukturen
- vgl. Gumm/Sommer 4.1 . . . 4.3

Suchen

- Spezifikation:
  - Eingabe: Folge $a = [a_1, \ldots, a_n]$ von Zahlen, Zahl x
  - Ausgabe: JA und ein i mit $x = a_i$, falls das existiert, sonst NEIN.

  Bsp: für $([4, 1, 3, 1], 1)$ sind JA 2 und JA 4 korrekt
- Algorithmus: falls $x = a_1$ dann JA 1, sonst: falls $x = a_2$, dann JA 2, sonst . . . falls $x = a_n$ dann JA n sonst NEIN
- Korrektheit: offensichtlich
- Komplexität: höchstens n Vergleiche, also linear
- geht besser? nein: dieser A. darf nicht früher NEIN sagen
- was muß man ändern? Eingaben einschränken

Suchen mit geordneter Eingabe

- Spezifikation:
  - Eingabe: Folge $a = [a_1, \ldots, a_n]$ von Zahlen, die schwach monoton steigt ($a_1 \leq a_2 \leq \cdots \leq a_n$), Zahl x
  - Ausgabe: JA und ein i mit $x = a_i$, falls das existiert, sonst NEIN.

  Bsp: $([0, 1, 4, 5, 5, 100, 3333], 2)$
- Algorithmus (Ansatz): $m = \lceil n/2 \rceil$, vergleiche x mit a_m
  - wie weiter? (Suchen in einer Teilfolge—welcher?)
  - wann aufhören? (wenn Folge kurz genug—zwei Fälle!)
- ist korrekt wegen Transitivität von $\leq$
- Komplexität: alle möglichen Abläufe in Entscheidungsbaum darstellen, maximale Laufzeit ist dessen Tiefe

Einfügen in geordnete Eingabe

- Spezifikation:
  - Eingabe: Folge $a = [a_1, \ldots, a_n]$ von Zahlen, die schwach monoton steigt ($a_1 \leq a_2 \leq \cdots \leq a_n$), Zahl x
  - Ausgabe: ein i, für das $[a_1, \ldots, a_{i-1}] \circ [x] \circ [a_i, \ldots, a_n]$ schwach monoton steigt ($\circ$ bedeutet Verkettung)

  Bsp: $([0, 1, 4, 5, 5, 100, 3333], 2)$, $([1, 2, 3], 0)$, $([1, 2, 3], 5)$
- Algorithmus: wie binäre Suche, aber nicht NEIN sagen, sondern zuletzt betrachtete Stelle i (oder $i + 1$) ausgeben
- offensichtl. korrekt, Komplexität: $\lceil \log_2(n + 1) \rceil$ Vergleiche
- Kosten der konkreten Ausführung hängen von konkreter Datenstruktur ab. Bsp: (alphabetisch) geordnetes Bücherregal, geordnete Datensätze in Datei

Sortieren

- Spezifikation:
  - Eingabe: Folge $a = [a_1, \ldots, a_n]$ von Zahlen
  - Ausgabe: eine schwach monoton steigende Permutation (Umordnung) von a

  d.h., kein Element geht veloren, keines wird erfunden
- Bsp: $[3, 1, 4, 1, 5, 9]$
- Sortieren ist klassisches Beispiel der Algorithmik,
- in der Praxis möchte man Monotonie (ermöglicht schnelle Suche), aber ohne (vollständiges) Sortieren (zu teuer): Ordnung bereits vorhandener Daten ausnutzen
- das motiviert den Algorithmus: Sortieren durch Einfügen

Sortieren durch Einfügen

- Algorithmus (mit Eingabe: $a = [a_1, \ldots, a_n]$)

  $b^{(0)} = []$; $b^{(1)} =$ Einfügen von a_1 in $b^{(0)}$; . . . ;
  $b^{(n)} =$ Einfügen von a_n in $b^{(n-1)}$; Ausgabe: $b^{(n)}$
- Korrektheit: wenn korrekter Algorithmus zum Einfügen benutzt wird, dann gilt die *Invariante*: für jedes i:
  $b^{(i)}$ ist monoton steigende Permutation von $[a_1, \ldots, a_i]$.
- damit erfüllt jedes $b^{(i)}$ die Voraussetzung für das Einfügen in *geordnete* Folge, also benutzen wir *binäres* Einfügen
- Komplexität: $\lceil \log_2(1) \rceil + \ldots + \lceil \log_2(n) \rceil \leq n \log_2 n$

  konkrete Werte für $n = 1, 2, 3, 4, 5, \ldots, 8$?

Die informationstheoretische untere Schranke für das Sortieren

- für jeden Sortier-Algorithmus für n Eingaben: alle möglichen Abläufe bilden Entscheidungsbaum (G/S 4.3.8)
  - in jedem inneren Knoten steht eine Frage ($a_i < a_j$), zwei Kinder sind Fortsetzungen bei Antwort Ja/Nein
  - in jedem Blatt steht eine Ausgabe (eine Permutation)

  Ü: dieser Baum für Sort. d. bin. Einf. für $n = 3$
- jede Permutation muß vorkommen, es gibt $n!$ (Fakultät) Permutationen, Höhe des Baumes (größte Anzahl innerer Knoten auf Pfad von Wurzel zu Blatt) $\geq \lceil \log_2(n!) \rceil$
- konkrete Zahlenwerte für $n = 1, 2, 3, 4, 5, \ldots, 8$
- vgl. mit Sortieren durch binäres Einfügen

Optimales Sortieren für 5 Eingaben

- Sort. d. binäres Einfügen: $1 + 2 + 2 + 3 = 8$ Vergleiche, informationsth. Schranke: $\lceil \log_2(5!) \rceil = \lceil \log_2(120) \rceil = 7$
- ein Verfahren mit 7 Vergleichen ist *merge insertion*:
  Formulierung mit Vergleich ($x < y$) und Tausch ($x \leftrightarrow y$)
  - Eingabe $[a, b, c, d, e]$
  - wenn $a > b$, dann $a \leftrightarrow b$; wenn $c > d$, dann $c \leftrightarrow d$
  - wenn $b > d$, dann $\{\ b \leftrightarrow d; a \leftrightarrow c\ \}$ (das geht nicht mit Sortiernetz) jetzt gilt $a \leq b \leq d \wedge c \leq d$ (Diagramm!)
  - e binär einfügen in $[a, b, d]$; c binär einfügen in ...

  Ü: Vergleiche nachzählen (wieviele für letztes Einfügen?)
- für beliebige n: Ford, Johnson 1959, zitiert in Knuth: *Art of Computer Programming Vol 3*, 1973

Hausaufgaben

1. Geben Sie ein Verfahren an, das die dritt-größte von fünf Zahlen (also den Median) durch höchstens 6 Vergleiche bestimmt.

   Sie können die Zahlen also nicht sortieren, denn das kostet 7 Vergleiche.

   Hinweis: trotzdem so anfangen wie bei Merge Insertion (bis „jetzt gilt“).

   Geben Sie alle Permutationen $[a, b, c, d]$ von $[1, 2, 3, 4]$ an, für die gilt $a \leq b \leq d \wedge c \leq d$

   Beschreibe Sie die Permutationen $[a, b, c, d, e]$ von $[1, 2, 3, 4, 5]$, für die gilt $a \leq b \leq d \wedge c \leq d$. Nicht alle hinschreiben, aber die Anzahl bestimmen und: Warum

kann d in einer solchen Permutation nicht der Median von $[1, 2, 3, 4, 5]$ sein?

Durch zwei weitere Vergleiche soll ein weiteres Element (aus dem gleichen Grund) ausgeschlossen werden.

Der letzte Vergleich führt zum Ziel.

2. In einem Behälter sind 91 Atome vom Typ A ... (Aufgabe gestellt in der ersten VL)

3. für dieses Programm:

```
let { m (x::Pic) = Transform Mirror x
    ; r (x::Pic) = Transform Rotate x
    ; d (x::Pic) = m (r x)
    ; a (x::Pic) = r (m x)
```



```
    ; pixel = Rectangle (10,10) True
    ; blank (p::Pic) = Combine Xor [p,p]
    ; sep (k::Nat)  =
      let { f (p::Pic) =  Row [ pixel, blan
      } in  iterate f k pixel
    ; f (k::Nat) (x::Pic) = Row
       [ Column [ x, sep k, d x]
       , a (Row [m (sep k), blank pixel, bl
       , m (Column [ x, sep k, d x])
       ]
} in f 3 (f 2 (f 1 (f 0 pixel)))
```

erklären Sie jede einzelne Funktion (rufen Sie sie einzeln auf), erklären Sie, wie dadurch das gesamte Bild entsteht.

Gezeichnet wird die *Hilbert-Kurve*: der weiße Weg berührt oder kreuzt sich nicht selbst und durchläuft das gesamte Quadrat. Begründen Sie diese Eigenschaften.

Können Sie das Programm verkürzen (bei genau gleicher Ausgabe)?

Hinweis: die Protokollierung der Auswertung von Teilausdrücken kann ab- und angeschaltet werden, indem `{-# silent #-}` oder `{-# verbose #-}` davorgesetzt wird:

```
...
} in {-# silent #-} f 3 (f 2 (f 1 ( {-# ver
```

Computernetze

Motivation, Überblick

- bisher betrachtet: Hard- und Software *eines* Computers
- tatsächlich benutzen viele Anwendungen (z.B. diese Online-Vorlesung) *mehrere* Computer gleichzeitig
- sind *Knoten* eines *Netzes* (das Internet), versenden entlang der *Kanten* des Netzes (z.B. Kabel, Funkstrecke) in *Protokollen* festgelegte Repräsentationen von Daten (z.B. Strom-Pulse, elektromagnetische Wellen) um bestimmte *Dienste* zu benutzen/zu erbringen
- Gumm/Sommer Kap. 7, 8; Kastens/Kleine-Büning Kap. 5
- technische u. soziale Fragen (*digitale Selbstverteidigung*)

Graphen (Definition, Beispiele)

- Def: ein Graph $G = (V, E)$ ist ein Paar aus Knotenmenge V und Kantenmenge E. Jede Kante $e \in E$ ist eine Teilmenge von V mit zwei Elementen.
- Bsp: $(\{a, b, c, d\}, \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}\})$
  verkürzte Notation für Kanten: $(\{a, b, c, d\}, \{ab, ac, bc, cd\})$.
  - der *leere* Graph auf $V = \{a, b, c, d\}$ ist $(V, \emptyset)$
  - der *vollständige* Graph (*Clique*) auf $V = \{a, b, c, d\}$
  - der *Pfad* auf $[c, a, d, b]$ ist $(\{a, b, c, d\}, \{ac, ad, db\})$
  - der *Kreis* auf $[c, a, d, b]$ ist $(\{a, b, c, d\}, \{ac, ad, db, bc\})$
- Def: die *Nachbarn* von x in G: $G(x) := \{y \mid xy \in E\}$,
  Def: der *Grad* von x in G: $\deg_G(x) = |G(x)|$.

Graphen (Anwendungen)

- Gütertransportnetze
  (Bsp: Knoten: Bahnhöfe, Kanten: Bahnstrecken)
  Ziel: schneller, sparsamer, ausfallsicherer Transport
- Datentransport (Rechnernetze)
  (Knoten: Rechner, Kanten: Kabel, Funkstrecke)
- das soziale Netz, die sogenannten sozialen Netzwerke
  (Knoten: Personen, Kanten: Beziehungen)
  Ziel: Erkennung der Person (aus Merkmalen des Endgerätes und des Verhaltens), Erfassung, Vorhersage und Steuerung ihres Verhaltens.

Zeichnungen von Graphen, Planarität

- Graph $G = (V, E)$ ist abstraktes mathematisches Objekt
- wird gern veranschaulicht durch eine Zeichnung:
  - jeder Knoten $v \in V$ ist Punkt der Zeichenfläche
  - jede Kante $xy \in E$ ist Linie von x nach y
- Vorsicht: wenn solche Linien kreuzen, ist das trotzdem kein Knoten von G. Deswegen Knoten geeignet markieren (dicke Punkte, kleine Kreise)
- Def: G heißt *planar* (deutsch: plättbar), wenn er eine kreuzungsfreie Zeichnung besitzt.
- Bsp: (Clique) K_4 ist planar, K_5 ist nicht planar

Isomorphie von Graphen

- Sind die Graphen $G_1 = (\{a, b, c, d\}, \{ab, bc, cd, ad\})$ und $G_2 = (\{1, 2, 3, 4\}, \{13, 14, 23, 24\})$ gleich?
- Nein, denn schon die Knotenmengen sind verschieden.
- Struktur stimmt überein: G_1 und G_2 haben gemeinsame Zeichnung, Unterschiede nur in Knoten-Namen
- Def: Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sind *isomorph*, Notation $G_1 \cong G_2$, falls es eine umkehrbar eindeutige Zuordnung $h : V_1 \to V_2$ gibt mit $\forall x, y \in V_1 : xy \in E_1 \iff h(x)h(y) \in E_2$.
- Bsp (forgesetzt) $h : a \mapsto 1, b \mapsto 3, \ldots$
- Anwendung: $G_1 \cong G_2 \cong C_4$, *der* Kreis mit 4 Knoten (eigentlich: *ein* . . .)

Zusammenhang von Graphen

- Def: $G_1 = (V_1, E_1)$ ist Teilgraph von $G_2 = (V_2, E_2)$, falls $V_1 \subseteq V_2$ und $E_1 \subseteq E_2$ (Teilmengen). Notation: $G_1 \subseteq G_2$
- Bsp: $(\{a, b, c\}, \{ab, ac, bc\}) \subseteq (\{a, b, c, d\}, \{ab, ac, bc, cd\})$.
- Def: $G = (V, E)$ heißt *zusammenhängend*, wenn für jede $x, y \in V$ ein Pfad P auf $[x, \ldots, y]$ Teilgraph von G ist.
- Bsp: für $V = \{a, b, c, d\}$: sind diese zusammenhängend? $(V, \{ab, cd\})$, $(V, \{ab, bc, ca\})$, $(V, \{ab, bd, cd\})$
- Anw.: G zshg $\iff$ man kann zwischen beliebigen Knoten Nachrichten senden (ggf. mittels anderer Knoten). Das ist oft das Ziel, aber nicht immer (Sicherheit!)

Beispiel für Weg im Internet-Graphen

- von einem Rechner in einem Computerpool der HTWK zu einem Rechner im Netz der Uni

```
$ traceroute www.uni-leipzig.de
traceroute to www.uni-leipzig.de (139.18.16.133), 3
 1  ciscof (141.57.11.253)  1.432 ms  1.624 ms  1.6
 2  fwintern.htwk-leipzig.de (141.57.196.13)  2.210
 3  141.57.193.4 (141.57.193.4)  3.304 ms  3.255 ms
 4  clusri-if.rz.uni-leipzig.de (141.57.252.17)  3.
 5  141.57.252.204 (141.57.252.204)  3.587 ms  3.55
 6  141.57.252.234 (141.57.252.234)  3.318 ms  1.92
 7  141.57.252.236 (141.57.252.236)  2.278 ms  3.23
 8  141.57.252.249 (141.57.252.249)  3.485 ms  3.46
 9  augate-if.rz.uni-leipzig.de (139.18.122.86)  5.
10  prod.uni-leipzig.de (139.18.16.133)  2.887 ms
```


Minimaler Zusammenhang: Bäume

- Def: G heißt *Baum*, wenn G zusammenhängend und kreisfrei ist (keinen Teilgraphen enthält, der ein Kreis ist)
- Bsp: der Pfad auf 4 Knoten
- jeder Term ist ein solcher Baum, bei dem zusätzlich die Wurzel und Reihenfolge von Nachbarn festgelegt wurde
- Def: $G \setminus e$ für $e \in E$: der Graph $(V, E \setminus \{e\})$
  (der Graph G ohne die Kante e)
- Satz: G Baum $\iff$ G kantenminimal zusammenhgd.
  (für alle $e \in E$ gilt: $G \setminus e$ ist nicht zusammenhängend)
- Satz: jeder Baum auf n Knoten hat $n - 1$ Kanten.
- Folgerung: das billigste zshg. Computernetz ist ein Baum

Redundanter Zusammenhang

- Motivation: Zusammenhang soll auch nach Ausfall einzelner Knoten bestehen und die (Um)wege dann auch automatisch gefunden werden
- Def: $G \setminus U$ für $U \subseteq V$: der Graph, der aus G ensteht durch Löschen aller Knoten in U und aller ihrer Kanten.
- Def: G heißt k*-fach (knoten)zusammenhängend*, wenn $|V| \geq k$ und für jede Menge U mit $|U| = k - 1$ gilt: $G \setminus U$ ist zusammenhängend.
  (also G einfach zshg. gdw. G zusammenhängend)
- Def: Zusammenhangszahl von G: das größte k, so daß G k-fach-zusammenhängend ist
- Bsp: für Pfad/Kreis/Clique auf 4 Knoten

Durchmesser von Graphen

- Motivation: schneller Transport durch kurze Wege
- Def: die *Länge* eines Pfades ist die Anzahl seiner Kanten.
  beachte: nicht . . . Knoten! Länge von $[a, b, c, d]$ ist 3.
- Def: der *Abstand* zweier Knoten x, y in G, Notation $\text{dist}_G(x, y)$, ist kürzeste Länge aller Pfade $[x, \ldots, y]$ in G.
- Def: der *Durchmesser* von G, Notation $\text{diam}(G)$, ist $\max\{\text{dist}_G(x, y) \mid x, y \in V\}$.
- Bsp: Durchmesser von Pfad/Kreis/Clique auf 4 Knoten
- Entwurfsziel bei Computernetzen (mit gegebenem V): kleiner Durchmesser (schnell) *und* wenige Kanten (billig) . . . *und* hoher Zusammenhang (ausfallsicher) und . . .

Ein Beispiel-Graph

- (der hat auch einen Namen, den möchte ich hier nicht verraten, damit Sie bei der Bearbeitung der Aufgaben nicht abgelenkt werden)
- die Knoten sind die Binärwörter der Länge 5 mit genau zwei Vorkommen der 1. (Bsp: $10100, 01001, \ldots$) – Wieviele?
- zwischen x und y besteht eine Kante genau dann, wenn x und y keine gemeinsame 1 haben.
  (Bsp: Kante zwischen $x = 10100 - 01001 = y$)
  Was ist $\deg_G(10100)$? Was ist $|E|$?

Der Hyperwürfel

- Hyperwürfel H_n der Dimension n ist Graph mit
  - $V_n = \{0, 1\}^n$ (alle Binärwörter der Länge n)
  - $E_n = \{xy \mid x$ und y unterscheiden sich in genau einem Bit $\}$
- H_1, H_2, H_3, H_4 zeichnen.
- Knoten- und Kanten-Anzahl von $H_1, H_2, H_3, \ldots H_n$
- H_n ist n-regulär (jeder Knoten hat Grad n)
- Zusammenhang, Durchmesser von H_n
- Anwendung: als Computernetz (für lokale Cluster)

Das Internet (Definition)

- jeder Knoten ist eine Netzwerkschnittstelle und besitzt eine IP-Adresse (Zahlenfolge)
- jede Schnittstelle gehört zu einem lokalen Netz (das ist meist Kreis, Pfad, Baum)
- Computer mit mehreren Schnittstellen (Gateway, Router, Bridge, Switch) verbinden solche lokalen Netze und bilden damit das globale Netz (Internet)
- jedes Datenpaket mit nicht-lokalem Ziel wird zu passendem Gateway gesendet, dieses sendet weiter

```
$ /sbin/route -n
Destination     Gateway         Genmask         Fla
0.0.0.0         141.57.11.253   0.0.0.0         UG
141.57.8.0      0.0.0.0         255.255.252.0   U
```


Internet und IP-Adressen

- der Laie meint mit „Internet“ das WWW (world wide web) = HTML-Dokument-Transport über HTTP(S)-Protokoll
- Internet 1972 (USA-Hochschulen), Arpanet 1969 (Militär)
- IP Version 4: Adresse besteht aus 4 mal 8 = 32 Bit, damit kann man $2^{32} \approx 4 \cdot 10^9$ Knoten adressieren
  diese Adressen sind ungleich verteilt und sehr teuer
- Router mit NAT (network address translation) verstecken Mehrfachbenutzung von Adressen in lokalen Netzen
- IP Version 6: Adresse besteht aus 8 mal 16 = 128 Bit
  jedes (private) Gerät hat feste (globale) Adresse?
  Privacy Extensions for . . . IPv6, 2007,
  `https://tools.ietf.org/html/rfc4941`

Firewalls, Virtuelle private Netze (VPN)

- Dienste können auf Teilnetze eingeschränkt werden, z.B. Zugriff nur aus Netz der HTWK, der Server selbst oder eine vorgeschaltete *Firewall* weisen andere Pakete aus anderen Netzen ab oder ignorieren sie
- durch VPN kann auf einem Rechner (z.B. Homeoffice) ein *virtuelle Netzwerkschnittstelle* mit Adresse in einem anderen Netz (z.B. Dienst-Netz) eingerichtet werden
- von der tatsächlichen Schnittstelle zum VPN-Provider besteht ein *VPN-Tunnel* (Pakete fahren vom Anwender zum VPN-Provider und erst von dort nach außen)
- der ISP (Internet Service Provider) des Anwenders sieht nur den verschlüsselten Verkehr zum VPN-Provider

Adressen und Namen

- niemand möchte/kann sich IP-Adressen von Rechnern/Servern merken (erst recht nicht IPv6)
- man benutzt stattdessen *Namen*, die durch DNS (domain name service) in Adressen übersetzt werden

```
$ cat /etc/resolv.conf    -- Konfigurations-Datei
nameserver 141.57.9.1     -- ein DNS-Server der HTW
$ nslookup tools.ietf.org -- Anfrage
Address: 4.31.198.61      -- Antwort
```

- am DNS-Server ist damit bekannt, welche IP-Adresse nach welchem Domain-Namen gefragt hat
- im Beispiel (von Rechner in HTWK) ist das egal, denn das Gateway (auch in HTWK) sieht dann auch das Paket, das an die erhaltene Adresse geschickt wird

Kampf um die (DNS-)Daten

- immer: der IP-Provider (und jedes Gateway unterwegs) leitet die Pakete weiter und kann die Adressen zusätzlich verwerten (den Inhalt nicht, der sollte verschlüsselt sein)
- die Überwachungswirtschaft will gern mehr wissen, deswegen bieten z.B. Google DNS-Server an (scheinbar kostenlos — also bezahlt mit Daten)
- Browser-Hersteller (z.B. Mozilla) empfehlen *DNS over HTTPS (DoH)*: Browser schickt dann DNS-Anfragen nicht an eigenes OS (und dieses unverschlüsselt an im OS eingestellten DNS-Server), sondern selbst verschlüsselt nach außen (z.B. Cloudflare, im Browser voreingestellt)

  . . . Verschlüsselung ist grundsätzlich gut und richtig?

Kampf um die (DNS-)Daten (Quelle)

- WSJ 29. Sept. 2019: Congressional antitrust investigators are scrutinizing plans by Google to use a new internet protocol [DoH] because of concerns that it could give the company a competitive advantage *by making it harder for others to access consumer data*.

  `https://www.wsj.com/articles/google-draws-house-antitrust-scrutiny-of-ir`
- also nicht etwa, weil die Daten der Verbraucher ausgeforscht werden, sondern weil eine Firma den anderen bei dieser Ausforschung zuvorkommt.

  das ist aber verständlich, denn das Bundeskartellamt ist für die Wirtschaft zuständig, nicht für den Datenschutz

Hausaufgaben

1. als Wiederholung zu Sortierverfahren:
   `https://www.mathekalender.de/wp/de/kalender/aufgaben/aufgabe-05/`

2. für $C_7 =$ ein Kreis mit 7 Knoten:

(a) Bestimmen Sie Durchmesser und Knotenzusammenhangszahl.
(b) Fügen Sie möglichst wenige Kanten ein, so daß der Durchmesser kleiner wird
(c) Fügen Sie möglichst wenige Kanten ein, um die Zusammenhangszahl zu erhöhen.

Für den Beispielgraphen (siehe Folie):

- geben Sie eine möglichst schöne Zeichnung an (regelmäßig, wenige Kreuzungen)
- bestimmen Sie den Durchmesser,
- bestimmen Sie die Zusammenhangszahl.

3. zu DNS und DoH (DNS over HTTPS) – Es geht bei diesen Fragen hier nur um die Technik, wir behandeln später noch die Überwachungs- und Anzeigenwirtschaft.

(a) Wo ist diese Einstellung in Firefox? Was ist das Default?
(b) Welche Information geht dem ISP bei DoH verloren? Die IP-Pakete sieht er ja dann doch beim Transport, egal, welcher Nameserver vorher verwendet wurde. Benutzen Sie diese Quelle: Timothy B. Lee *Why big ISPs aren't happy about Google's plans for encrypted*

DNS, Ars Technica, 10/1/2019. `https://arstechnica.com/tech-policy/2019/09/isps-worry-a-new-chrome-feature-will-stop-`

(c) Wie wirkt ein *Pi-Hole* als DNS-Server? Benutzen Sie Original-Quellen (`https://docs.pi-hole.net/`) und Sekundärquellen (z.B. Troy Hunt: *Mmm... Pi-hole...*, 26. Sept. 2018 `https://www.troyhunt.com/mmm-pi-hole/`) Welche Datenverarbeitung Ihrer Daten durch Dritte findet auf diesen Webseiten statt? Mit umatrix anzeigen, dann blockieren. Webseiten trotzdem noch benutzbar?
(d) Erklären und diskutieren Sie: wenn man uMatrix benutzt, braucht man das Pi-Hole braucht eigentlich

nicht?
Das Pi-Hole als DNS-Server für das gesamte lokale Netz verhindert auch die unerwünschte Datenübertragung von Geräten (z.B. sogenannten Smart-Fernsehern), deren Hersteller die Konfiguration durch den Besitzer verbieten. Es sei denn, das Gerät benutzt auch DoH.

Kodierung und Kompression

Kodierung: Definition,Motivation

- Kodierung: das Übersetzen einer Nachricht mit Alphabet Σ in eine Nachricht mit Alphabet Γ (ersetzt jedes Zeichen durch ein Wort) $c : \Sigma^* \to \Gamma^*$
  Dekodierung: $d : \Gamma^* \to \Sigma^*$, so daß für alle $w \in \Sigma^* : d(c(w)) = w$.
- häufig $\Gamma = \{0, 1\} = \mathbb{B}$ (physikalisch realisiert)
- Redundanz: Fehler erkennen und reparieren
- Effizienz: minimiere Länge von $c(w)$
- abweichende Wortbedeutungen für *Code*: Geheimsprache, Computersprache

Der Hamming-Abstand

- Menge der Code-Wörter $\mathbb{B}^n$ (Bit-Folgen der Länge n)
- der Hamming-Abstand von $u, v \in \mathbb{B}^n$ ist die Anzahl der Positionen i, an denen sich die Werte (Bits) unterscheiden. dist$(u, v) := |\{i : u_i \neq v_i\}|$
- Beispiele: dist$(0010, 1000)$, dist$(0101, 1010)$,
- benannt nach Richard Wesley Hamming (1915–1998) `https://www-history.mcs.st-andrews.ac.uk/~history/Mathematicians/Hamming.html`
- Eigenschaften: dist$(u, v) = 0 \iff u = v$, dist$(u, w) \leq$ dist$(u, v) +$ dist(v, w)
- Motivation: Übertragung $u \longrightarrow u'$ über fehlerhaften Kanal (ändert max. f von n Bits) liefert dist$(u, u') \leq f$

Abstände und Codes

- Def: die Hamming-Weite der Menge $M \subseteq \mathbb{B}^n$ ist
  dist$(M) = \min\{$dist$(u, v) : u \in M, v \in M, u \neq v\}$
  (der kleinste Abstand zwischen zwei Wörtern)
- Beispiel: dist$\{110, 101, 011\} = 2$.
- Finde möglichst viele Code-Wörter der Länge n mit Weite $\geq w$! (Beispiel: $n = 6, w = 3$)

Fehler-Erkennung und -Korrektur

- Ein Code M kann k-Bit-Fehler *erkennen*, falls dist$(M) > k$.
- Ein Code M kann k-Bit-Fehler *korrigieren*, falls dist$(M) > 2k$.
- Zu empfangenem (möglicherweise verfälschtem) Codewort w' bestimmt man das nächstliegende (bzgl. dist) tatsächliche Codewort.
- Falls dist$(M) > 2k$ und höchstens k Bit falsch, dann ist dieses eindeutig bestimmt.
- Wieviel Bits braucht man für eine Codierung der Dezimalziffern, die 2-Bit-Fehler korrigieren kann?

Anwendung, Ausblick

- wenn Fehlerrate des Kanals beschränkt ist,
  - fehler-korrigierenden Code benutzen
  - oder fehler-erkennenden Code benutzen und fehlerhafte empfangen Codewörter erneut senden
- die Fehler-Erkennung kann stattfinden:
  - für jedes einzelne Zeichen (wie bisher beschrieben)
  - für die gesamte Nachricht, durch Anhängen einer Prüf-Nachricht (Hash) nächste VL im Zshg. mit Signatur und Verschlüsselung
- jetzt: Kompression (sinnwahrende Verkürzung)

Kompression

- kürzere Darstellung von Daten unter Ausnutzung
  - der Struktur der Daten, z. B.
    - Anzahlen von Zeichen
    - Wiederholungen von Teildaten
  - der Art der Anwendung (Archivierung, Unterhaltung)
- man unterscheidet
  - verlustfreie K. (z. B. für Texte, Programme)
  - verlustbehaftete K. (z. B. Bild, Musik, Film)
- Anforderungen: (sind teilweise widersprüchlich)
  - hohe Kompression
  - schnelle Kompression
  - schnelle De-Kompression

Codes variabler Länge

- Abbildung $c : \Sigma \to \Gamma^*$ (von Buchstabe auf Wort)
  Bsp: $\Sigma = \{A, B, C, D\}$, $\Gamma = \{0, 1\} = \mathbb{B}$,
  $c = \{A \to 0, B \to 10, C \to 11, D \to 110\}$
- wird erweitert auf $c : \Sigma^* \to \Gamma^*$ (von Wort auf Wort) durch $c(x_1 x_2 \dots x_m) = c(x_1)c(x_2) \dots c(x_m)$
  Bsp: $c(CDB) = 1111010$
- heißt *Code*, wenn für alle $u, v \in \Sigma^*$: $c(u) = c(v) \Rightarrow u = v$
  Bsp: $u = CDB, v = CCAB$
- – $A \to 00, B \to 01, C \to 10, D \to 11$ ist Code
  – $A \to 00, B \to 0011, C \to 10, D \to 01$ ist . . . ?
  – $A \to 00, B \to 001, C \to 10, D \to 01$ ist . . . ?

Präfix-Codes (eigentlich: präfixfreie Codes)

- Def: $u \sqsubseteq v$ falls u Anfangsstück von v, Bsp: $00 \sqsubseteq 0011$
- $c : \Sigma \to \Gamma^*$ heißt *präfix-frei*, wenn für alle $x \neq y : c(x) \not\sqsubseteq c(y)$
- Satz: wenn c präfixfrei ist, dann ist c ein Code
- Beispiele:
  - $A \to 00, B \to 01, C \to 10, D \to 11$ ist ein Präfixcode.
  - $A \to 0, B \to 10, C \to 110, D \to 111$ ist ein Präfixcode.
  - $A \to 00, B \to 0011, C \to 10, D \to 01$ ist kein Präfixcode, aber ein Code, denn Spiegelbild ist Präfixcode.
  - Ü: ein Code c, der kein Präfixcode ist und sein Spiegelbild auch nicht

Effiziente Präfix-Codes

- Kosten für Präfix-Codes über $\{0, 1\}$
  für eine Nachricht w mit insgesamt 150 Zeichen,
  davon 10 A, 100 B, 30 C, 10 D:
- Bsp. $c : A \mapsto 00, B \mapsto 01, C \mapsto 10, D \mapsto 11$: $|c(w)| = 300$
- besser: B häufig $\rightarrow c(B)$ kurz, also $c(B) = 0$
  dann müssen die anderen Codewörter mit 1 anfangen
  $c : A \mapsto 1 \ldots, B \mapsto 0, C \mapsto 1 \ldots, D \mapsto 1 \ldots$
- für jede Verteilung von Zeichen kann man einen optimalen Präfixcode bestimmen (D. A. Huffman, 1952)
  Anwendung z.B. im Bildformat PNG (P. Deutsch, 1996)
  `https://www.ietf.org/rfc/rfc1951.txt`

Kompression durch Blockwiederholung

- jedesmal, wenn in der Eingabefolge eine (genügend lange) Teilfolge wiederholt vorkommt,
  dann wird diese nicht ausgegeben, sondern durch einen Verweis ersetzt (zwei Zahlen: Verschiebung und Länge).
  (Abraham Lempel, Jacob Ziv, 1977)
- Dekompression ist einfach und schnell.
  der Kompressor muß Wiederholungen finden, aber nicht unbedingt alle (dann schneller, aber Ausgabe größer)
- Anwendung: auch in PNG.
  gut für Grafiken (Bsp: in Zeichenaufgabe im autotool)
  nicht Fotos (keine exakten Wiederholungen in der Natur)

Beispiele für verlustfreie Kompression

- das Bild `Circle 800`
  $800^2 = 640.000$ Pixel (Bit) = 80 kByte.
  als PNG: 5052 Byte.
  tatsächlich genügen 10 Byte (für den Programmtext) — wenn man dazu das autotool hat, um ihn auszuführen
- alle (127) Folien diese VL bis jetzt
  Latex-Quelltexte: 107.660 B, komprimiert (`zip`): 36.681 B
  PDF: 305.368 B, komprimiert (`zip`): 291.798 B
  (denn PDF enthält bereits komprimierte Daten)

Kompression durch Geradeaus-Programme

- (Wiederholung) unsere Programmiersprache mit `let`-Bindungen für Namen, ohne Unterprogramme
- (neu) semantischer Bereich: Folgen (anstatt Bilder), Operation: Verkettung von zwei Folgen (vgl. `Row[x,y]`)
- ```
  let { a = Singleton 0 ; b = Singleton 1
      ; c = Append a b ; d = Append c c
      ; e = Append d b; f = Append c e } in f
  ```
- Beziehung zu Lempel-Ziv-Verfahren:
  jeder Name bezeichnet eine wiederholbare Teilfolge
- Auswertung des Programms = Dekompression,
  Finden eines (kurzen) Programms mit gegebenem Wert = Kompression (dazu Hausaufgabe in autotool)

Verlustbehaftete Kompression

- Anwendung/Beispiel: Ton (MP3), Bild (JPEG), Film (MP4)
- Daten aus natürlichen Quellen sind kaum kompressibel
- Auge, Ohr, Gehirn nehmen nicht alle Details wahr: Kompression kann Details verfälschen oder entfernen
- typische Auswirkungen solcher Verfahren:
  - schlechter Ton (fehlende oder verschmierte Höhen) bei: Leipziger Straßenbahn-Ansagen, Videos, Internet-Radiostationen (die nur Video-Ton abspielen) aber: Umgebungsgeräusche, schlechte Ohrhörer
  - Artefakte in stark vergrößerten Bild-Ausschnitten
  - plötzliche Unschärfen bei detailreichen Film-Szenen (bewegte Wasserfläche, Gras, Wald)

Beispiel Video-Kompression

- eine *Tagesschau*
  15 min, 25 Bilder pro Sekunde, also 24.000 Bilder
  HD: 1280 mal 720 Pixel $\approx 10^6$ Pixel
  je Pixel 3 Byte Farbe: 3 MB je Bild, gesamt: 72 GB
  Format mp4 (Codec H.264): 218 MB. Das sind 0.3% !
- Kompression nach H.264 benutzt
  - inter coding: zeitlich statistische Abhängigkeiten zwischen verschiedenen Bildern
  - intra coding: räumliche stat. Abh. innerhalb eines Bildes

  `https://www.itu.int/rec/T-REC-H.264-201906-I/en`

Bespiele Audio-Kompression

- Für ein rohes (nicht komprimiertes) Stereo-Audio-Signal wird der Amplitudenverlauf jedes Kanals mit 48 kHz abgetastet und jeder Wert als Zahl mit 16 Bit gespeichert.
  Welche Bitrate (Bit pro Sekunde) hat dieses Signal?
  Lösung: pro Sekunde: 2 Kanäle mal $4.8 \cdot 10^4$ Samples mal 16 Bit je Sample $= 1.5 \cdot 10^6$ Bit.
- Bestimmen Sie die Bitraten von bluemont.{ogg,ac3}
  `https://gitlab.imn.htwk-leipzig.de/waldmann/computer-mu/-/tree/master/tidal/code`
  Lösung:
  - ogg: 760 kByte/1 min $= 6 \cdot 10^6$bit$/60$s $= 10^5$bit/s.
  - ac3: halbe Dateigröße $\Rightarrow$ halbe Bitrate

Hausaufgaben

1. Zeichnen Sie einen Hyperwürfel H_3 der Dimension 3 (siehe Folie bei Graphen)
   Geben Sie eine möglichst große Teilmenge C_1 der Knoten an, die einen Code bildet, der 1-Bit-Fehler erkennen kann.
   Geben Sie eine möglichst große Teilmenge C_2 der Knoten an, die einen Code bildet, der 1-Bit-Fehler reparieren kann.
   Desgleichen für H_4. (Hinweis: es ist unpraktisch, und auch nicht nötig, diesen Graphen zu zeichnen.)
   Bestimmen Sie eine obere Schranke für die Anzahl der Elemente eines 1-Bit-Fehler-reparierenden Codes C im
  ```

H_4:

Für jeden Knoten $x \in C$ betrachten wir die Menge $H_4[x] = \{x\} \cup H_4(x) =$ der Knoten x mit seinen Nachbarn.

Jede solche Menge hat ... Elemente (warum?)

Alle diese Mengen sind disjunkt (warum?)

Die Vereinigung dieser Mengen liegt in $V(H_4)$. Also gibt es höchstens ... solche Mengen.

2. Geben Sie eine Darstellung dieser Folge der Länge 100 durch ein möglichst kurzes Geradeaus-Programm an.

$$s = 1001000010000001000 \ldots 0001$$

Die 1 stehen auf den Indizes, die Quadratzahlen sind.

Verallgemeinern Sie a) auf längere Folgen dieser Art, b) auf entsprechende Folgen für Kubikzahlen.

3. `https://www.mathekalender.de/wp/de/kalender/aufgaben/aufgabe-06/`

Modellierung: G ist Kreis C_{33} auf Knoten 1, 2,..., 33 mit zusätzlichen Kanten (Diagonalen)
$1-16, 1-17, 2-17, 2-18, \ldots$.

Wieviele Kanten hat G? Welchen Grad hat jeder Knoten aus G? Welchen Durchmesser hat G?

Gesucht ist eine größte unabhängige Knotenmenge U (rote Wichtel).

Def. eine Menge $U \subseteq V$ heißt *unabhängig in* $G = (V, E)$, falls $\forall x, y \in U : xy \notin E$.

Aufgabe kann ersetzt werden durch andere Aufgabe aus dem gleichen Kalender, bei der Graphen vorkommen oder zur Lösung benutzt werden können. Rechtzeitig anmelden!

Hinweise und Diskussion im Forum des Opal-Kurses (und nicht hier — die Aufgabensteller wünschen keine öffentliche Diskussion, denn der Wettbewerb der angemeldeten Teilnehmer soll nicht verzerrt werden.)

Verschlüsseln und unterschreiben

Motivation, Überblick

- Verschlüsseln: verhindert unbefugtes Mitlesen von Nachrichten, die
  - über öffentlichen Kanal (z.B. Internet) gesendet,
  - auf fremden Rechnern (sog. Cloud) gespeichert werden
- Unterschreiben (signieren):

  zur Bestätigung der Identität des Absenders (z.B. Webserver) einer Nachricht (Prüfung der Unterschrift)

  und der Integrität der Nachricht (feste Verbindung von Nachricht und Unterschrift)

Verschlüsseln: Definition, Beispiel

- K Klartexte, S Schlüsseltexte, G Geheimnisse
  verschlüsseln (encrypt): $e : G \times K \to S$
  entschlüsseln (decrypt): $d : G \times S \to K$,
  so daß $d(g, e(g, k)) = k$
- Bsp: $s_2 : A \mapsto C, B \mapsto D, \ldots, X \mapsto Z, Y \mapsto A, Z \mapsto B$
  das Geheimnis ist die Verschiebung (hier 2)
  entschlüsseln mit s_{-2} (Zurückschieben), $s_{-d}(s_d(k)) = k$
- Geheimnis ist leicht zu erraten bei bekannter Verteilung von K (Bsp: *e* ist häufigster Buchstabe im Deutschen)
  Ausweg: längere Buchstabengruppen zusammenfassen
- grundsätzliche Eigenschaft bleibt: benötigt sicheren Kanal zur Übertragung des Geheimnisses

Verfahren mit öffentlichen Schlüsseln

- Absender (Alice), Empfänger (Bob), keine gemeinsames Geheimnis, offener (abgehörter) Kanal
- trotzdem sicherer Versand nach RSA-Verfahren (Ron Rivest, Adi Shamir, Leonard Adleman, 1977)
- – B schickt offenes Vorhängeschloß zu A,
  – A verschließt Kiste damit und schickt an B
  – B öffnet mit Schlüssel

  das Schloß = der *öffentliche* Schlüssel (z.B. auf Webseite publiziert), eigentlicher Schlüssel ist *privat*
- Rechnen mit Restklassen, Satz von Fermat (1607–1665)
- (später) B kann nicht sicher sein, daß Nachricht von A kommt (jemand kann unterwegs das Schloß klauen)

Das Rechnen mit Restklassen

- Def: $[a]_m =$ der Rest bei Division von a durch m
  wenn $a = q \cdot m + r$ mit $0 \le r < m$, dann $[a]_m = r$
  Bsp: $[11]_4 = 3$, denn $11 = 2 \cdot 4 + 3$; $[100]_{11} = \ldots$; $[20]_7 = \ldots$
  Def: $x \equiv_m y$ falls $[x - y]_m = 0$. Bsp $87 \equiv_4 11$
- Satz: $[a + b]_m \equiv_m [a]_m + [b]_m$, Bsp. $m = 4, a = 11, b = 6$
- Satz: $[a \cdot b]_m \equiv_m [a]_m \cdot [b]_m$, Bsp. $m = 4, a = 11, b = 6$
- Satz: $[a^b]_m \equiv_m [a]_m^b$, Bsp. $m = 4, a = 5, b = 7$
- im Exponenten? $[a^b]_m \not\equiv_m a^{[b]_m}$ für $m = 5, a = 2, b = 6$
  der (kleine) Satz von Fermat beschreibt das genauer

Restklassen von Potenzen

- Satz von Fermat (Spezialfall): für alle Primzahlen $p \neq q$, für jede Zahl k teilerfremd zu pq, gilt $[k^{(p-1)(q-1)}]_{pq} = 1$
- Beispiel: $p = 3, q = 11, k = 2$:
  $[2^{2 \cdot 10}]_{33} = [1.048.576]_{33} = [31.775 \cdot 33 + 1]_{33} = 1$
- wenn (e, d) mit $de \equiv_{(p-1)(q-1)} 1$, Bsp. $e = 7, d = 3$
  also $ed = f \cdot (p-1)(q-1) + 1$, Bsp. $21 = 1 \cdot 20 + 1$
  dann $k^{ed} = k^{f(p-1)(q-1)+1} = k^{(p-1)(q-1)f} \cdot k =$
  $(k^{(p-1)(q-1)})^f \cdot k \equiv_{pq} 1^f \cdot k = k$
  Bsp: $k^{ed} = 2^{21} = 2^{20+1} = 2^{20} \cdot 2^1 \equiv_{33} 1 \cdot 2$,
- im folgenden: Klartext k, Schlüsseltext $s = k^e$,
  Entschlüsselung $s^d = (k^e)^d = k^{ed} \equiv_{pq} k$

Das RSA-Verfahren

- wer Nachrichten empfangen möchte, wählt $p \neq q$ prim, e, d mit $ed \equiv_{(p-1)(q-1)} 1$, hält d geheim und veröffentlicht: e und pq (das Produkt, aber nicht die Faktoren)
- der Absender einer Nachricht k verschlüsselt: $s = [k^e]_{pq}$,
- der Empfänger entschlüsselt $[s^d]_{pq} = [k^{e \cdot d}]_{pq} \equiv_{pq} k$
- dabei pq so groß (z.B. 1000 Dezimalstellen), daß die Berechnung von p, q und d (privater Schlüssel) aus den veröffentlichten Daten praktisch unmöglich ist
  Bsp: $pq = 10^{28} + 3 = 813219713 \cdot 1229679979486675331$
- Angriff: alle Nachrichten speichern in der Hoffnung auf besser Algorithmen/schnellere Rechner in der Zukunft

Rechnen mit großen Zahlen

- bei Ver- und Entschlüsseln: effiziente Berechnung von $[a^b]_c$ für $a, b, c \approx 10^{1000}$ mit diesen Tricks:
- das Potenzieren, Bsp: 3^6
  - nicht als iteriertes Multiplizieren, $3^6 = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$
  - sondern durch Geradeausprogramm mit Multiplizieren und Quadrieren, Bsp: $3^6 = (3^3)^2 = (3^2 \cdot 3)^2$

  Ü: solche Formeln (als Programme) für a^{13}, a^{111}
- die Bestimmung der Restklasse $[\ldots]_c$

  nicht erst zum Schluß (Zwischenergebnisse zu groß),

  sondern nach jeder einzelnen arithmetischen Operation

Anwendung: OpenPGP message format

- `https://tools.ietf.org/html/rfc4880`
  - A creates a message m
  - A generates a random number g as session key
  - A encrypts g using B's public key. ($[g^e]_{pq}$)
  - A encrypts m using g (mit einfachem Verfahren)
  - B decrypts $e(g)$ using her private key ($[g^{ed}]_{pq}$)
  - B decrypts $e(m)$ using g (einfaches Verfahren)
- z.B. in Thunderbird `https://support.mozilla.org/en-US/kb/introduction-to-e2e-encryption`
- privater Schlüssel und dekodierte Nachrichten *niemals* außerhalb des eigenen Rechners
- verbleibende Aufgaben: Bekanntmachung und Verifikation der öffentlichen Schlüssel.

Verschlüsselte Mails und Überwachung

- Überwachungswirtschaft bevorzugt Webmailer (dort Verkauf von Werbeplatz) anstatt lokaler Mail-Clients
- in-Browser-Programme dürfen nicht auf lokale Daten zugreifen, aber privater Schlüssel muß lokal sein
- möchte gern auch Mail-Inhalte mitlesen (genaueres Personenprofil für zielgerichteten Anzeigenverkauf)
  (ansonsten eben Kostruktion des sozialen Graphen aus den Transport-Daten, wenn Inhaltsdaten verschlüsselt)
- (verschlüsselte) Mails mit HTML-Code, der auf externe Ressourcen verweist, (z.B. Bilder, auch unsichtbare, Tracking-Pixel): Datentransport beim Anclicken oder beim automatischen Laden ($\Rightarrow$ abschalten!)

Authentizität (Identitäts-Beweis)

- bisher gezeigte Verfahren gewährleisten *Vertraulichkeit* (nur Empfänger kann Nachricht entschlüsseln)
- weiteres Ziel ist *Authentizität* (Nachricht stammt vom angegeben Absender)
- für Authentizität beweist Absender A, daß er den privaten Schlüssel der Identität A besitzt, indem er ihn benutzt
- dazu verschlüsselt A eine (bekannte) Nachricht m (z.B. die Schlagzeile einer bestimmten Tageszeitung) mit seinem privaten (!) Schlüssel
- B entschlüsselt diese mit A's öffentlichem Schlüssel und testet, daß das Resultat m ist.
- das funktioniert, weil $(m^d)^e = m^{de} = m^{ed} = (m^e)^d$

Authentizität und Integrität von Nachrichten

- soll nachweisen, daß bei B ankommende Nachricht tatsächlich genau so von A gesendet wurde.
- benutzt Verfahren von vorige Folie, die „bekannte Nachricht“ ist jetzt die (verschl.) übertragene Nachricht.
  `https://tools.ietf.org/html/rfc4880#section-2.2`
- tatsächlich wird eine Zusammenfassung (digest, hash code) $h(m)$ der Nachricht m signiert,
  Dabei soll $h(m)$ (wesentlich) kürzer als m sein (damit das Signieren schneller geht)
  aber trotzdem fälschungssicher: zu gegebenem m findet man praktisch kein m' mit $m \neq m'$ und $h(m) = h(m')$

Zertifikats-Ketten, Webserver-Zertifikate

- das (qualifizierte) Zertifikat eines Webservers nennt die verantwortliche (juristische) Person (z.B. Fakultät)
- diese Angabe ist signiert (nach dem vorigem Verfahren) durch eine zertifikatausgebende Stelle (z.B. ITSZ HTWK im Auftrag des DFN) nach tatsächlicher Identitätsprüfung (persönliches Erscheinen, Ausweis zeigen)
- Identität dieser Stelle ist signiert . . . (evtl. viele Schritte) bis schließlich zu einem *root certificate*, von dessen Gültigkeit man sich auf anderem Weg überzeugen muß.
- Browser haben einige Root-Zertifikate fest installiert.
- Zertifikate von Letsencrypt bescheinigen nur, daß man unter angegebem Domain-Namen einmal erreichbar war.

Transportverschlüsselung (TLS)

- TLS (transport layer security)
  `https://tools.ietf.org/html/rfc8446`
  zur Verschlüsselung von Datentransporten im Internet (Paket-Absender, Empfänger, Länge sind sichtbar, Inhalt ist verschlüsselt)
- u.a. bei Protokoll HTTPS (hypertext transport protocol, secure) zwischen Web-Server und Client (Browser)
- Vereinbarung (ohne bekannte öffentliche Schlüssel) eines einmaligen gemeinsamen Geheimnisses g, dann Transport mit einfachem Code, der g benutzt
- TLS ist unabhägig vom Web-Server-Zertifikat und wird auch für andere Dienste benutzt

Hausaufgaben

WS21: Aufgaben 1, 3, 4.

1. Geben Sie Sie ein möglichst kurzes Geradeausprogramm an (vgl. Aufgabe zu Kompression einer Zeichenkette), das a^{111} nur mit Multiplikationen bestimmt.

   Etwa $b = a \cdot a, c = b \cdot b, \ldots$

   Benutzen Sie die Binärdarstellung des Exponenten.

   Verallgemeinern Sie dazu dieses Beispiel:
   $13 = 2 \cdot 6 + 1 = 2(2 \cdot 3 + 0) + 1 = 2(2(2 \cdot 1 + 1) + 0) + 1 = 2(2(2 \cdot 0 + 1) + 1) + 0) + 1$,
   also $a^{13} = a^{2 \cdot 6 + 1} = (a^6)^2 \cdot a$ usw.

   Berechnen Sie damit $[3^{13}]_5, [3^{111}]_5$.

2. Welche Fälschung wird ermöglicht, wenn bei dem auf Folie *Authentizität und Integrität von Nachrichten* angegebenem Verfahren eine Hashfunktion h benutzt wird, für die man zu gegebenem m leicht andere m' findet mit $h(m) = h(m')$?

   Begründen Sie, daß folgende Funktionen diese Eigenschaft haben (und deswegen ungeeignet sind)

   (a) $h_1(m) = 0$
   (b) $h_2(m) =$ Länge (Anzahl der Zeichen) von m
   (c) $h_3(m) =$ Summe der ASCII-Codes der Zeichen von m

3. Zeigen Sie Zertifikat und Zertifikatskette für `htwk-leipzig.de` (und ähnliche, z.B. autotool, bildungsportal-sachsen.de) Welche Bitbreite wird verwendet? Wieviele Dezimalziffern wären das?

   Welches Root-Zertifikat wird verwendet, warum glaubt das Ihr Browser?

   Fassen Sie wesentliche Inhalte dieser Meldung über die (2019 zeitweilig) erzwungene Installation von Root-Zertifikaten in Kasachstan zusammen:

   Jon Brodkin: *Google, Apple, and Mozilla block Kazakhstan government's browser spying*, Ars Technica 21. 8. 2019, `https://arstechnica.com/tech-policy/2019/08/chrome-firefox-and-safari-updated-to-block-`

   Was ist der aktuelle Status?

4. Welche Daten sind im digitalen Impfzertifikat enthalten? Wie wird die Gültigkeit eines Zertifikates für eine konkrete Person geprüft? Warum ist das Verfahren fälschungssicher?

   Hinweis: die Antwort ist nicht: *da muß man eben die richtige App verwenden*. Die Form der Daten und ihre Gültigkeit ist unabhängig von einer konkreten Software spezifiziert. Der Gesetzgeber kann nicht die Installation einer bestimmten Software oder überhaupt Besitz und Benutzung eines sogenannten Smartphones vorschreiben.

   Zitieren Sie Primärquellen, z.B.

   - *EU Digital COVID Certificate*, `https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_en`
   - *European eHealth network - Digital Covid Certificate* `https://github.com/ehn-dcc-development/hcert-spec`
   - Sächsische Corona-Notfall-Verordnung (28. 12. 2021)

   Sie können Sekundärquellen verwenden, um die Primärquellen zu finden und zu verstehen.

   Vorsicht: Geben Sie Ihr eigenes Zertifikat (während der Bearbeitung dieser Aufgabe und auch sonst) nicht ohne triftigen Grund an Dritte weiter (z.B. eine angeblich

   Online-Zertifikatsprüfung). Fälschen kann man es nicht (siehe Teilaufgabe oben), aber es enthält doch personenbezogene Daten, an denen die Überwachungswirtschaft interessiert ist.

   Welche Vorschriften machen die Gesetzgeber (in EU, Bund, Sachsen) über die Verarbeitung personenbezogener Daten (z.B. durch Gastronomen) bei Impfnachweis (u.ä.) und Kontakterfassung?

5. (optional) Schicken Sie mir (bis vor der nächsten Übung) eine nach OpenPGP verschlüsselte Email (von Ihrer Hochschuladresse aus). Inhalt `test` reicht.

   Wenn Sie Ihren public key mitschicken, werde ich verschlüsselt antworten. (Auch ohne Inhalt.)

   Vorsicht: Ihren public key *nicht* auf Keyserver hochladen. Keyserver sind eine Ansammlung von personenbezogenen Daten. (Lebende Email-Adressen—sind viel wert. Von Leuten, die sich mit Technik auskennen—sind noch viel mehr wert.)

   Welche Vorkehrungen zu Datenschutz/Datenreduzierung sehen Sie auf `https://keys.openpgp.org/`? (ist voreingestellter Keyserver bei Thunderbird)

Protokolle, Dienste

Überblick

- Dienst: Datenverarbeitung (Dienstleistung) auf einem (fremden) Computer (Server, „Diener“)
- Protokoll: beschreibt Form der Daten und Art ihrer Übertragung zw. Auftraggeber (Kunde, Client) und Server
  - transport-spezifisch (Bsp: Internet-Protokoll TCP/IP)
  - dienst-spezifisch (Bsp: Email-Transport)
- Protokolle: offen oder geschlossen (geheim)
- Dienstleistung: dezentral (föderiert) oder zentralisiert
- Bezahlung der Leistung: direkt oder verdeckt
- Gumm/Sommer Kap. 8.5 Dienste im Internet

Dienste: lokal, entfernt, zentral?

- um eine Aufgabe mit Computerhilfe zu lösen:
  - Computer selbst entwerfen? bauen? betreiben? mieten? (im eigenen Haus, im fremden?)
  - Software: schreiben? installieren? warten? mieten?
- Bsp: autotool: auf virtueller Maschine im ITSZ der HTWK
- Bsp: HTWK bezahlt BPS GmbH für Betrieb von Opal, BPS bezahlt Personal und mietet Hardware
- vergleiche: eigener Brunnen? Ofen? Generator? Auto?
- Einkauf von Dienstleistungen kann ökonomisch sein,
- Zentralisierung (Monopolisierung) auch, wird dann aber staatlich geregelt (Infrastuktur: z.B. Straßenbau)

Lokale Dienste, Betriebssysteme

- bereits jeder einzelne Computer enthält verschiedene Hard- und Softwarekomponenten
- deren gemeiname Schnittstelle ist das *Betriebssystem*, es teilt die Hardware-Ressourcen (Rechenzeit, Hauptspeicher, Plattenspeicher, Ein/Ausgabegeräte) den Dienst- und Benutzerprogrammen zu
- OS/360 1964 (Stapelbetrieb, geräteunabhängige E/A) UNIX 1970 (multi-user, multi-process), GNU (Gnu's not Unix) 1983, GNU/Linux 1991, Android 2007
  CP/M 1974, MS-DOS 1980, MS-Windows 1983
- Gumm/Sommer Abschnitt 1.6, Kapitel 6

Beispiele für Netzwerk-Dienste: DNS, NTP

- Namens-Auflösung (DNS, Domain Name System)
- Zeit (NTP, Network Time Protocol)
  `https://tools.ietf.org/html/rfc5905`
  - übereinstimmende und exakte Uhren: Voraussetzung für die Funktion anderer Dienste in Netzwerken
  - Ü: welchen Zeit-Server benutzt Ihr Computer? Mglw. den des Routers, was ist dort eingestellt? (D.h., wer kann die Daten wirtschaftlich verwerten)
  - Wie genau geht die Uhr Ihres Computers?
    `https://time.gov/` (your clock is off by ...)
    Wie wird das gemessen? (beachte Entfernung und Lichtgeschwindigkeit)

Electronic Mail (Email)

- Email 1971, SMTP (simple mail transfer protocol) 1982
  Nachricht besteht aus: *Kopf*: Empfänger, Absender, *Inhalt*: Text, auch anderen Medien (kodiert als Text) (MIME, multipurpose internet mail extension)
- jeder Rechner im Internet kann im Prinzip Mails senden und empfangen
- beliebig senden: Spam, deswegen Weiterleitung nur von etablierten Quellen
- der Empfangs-Rechner muß immer online sein, also nicht eigener PC/Notebook, sondern Trennung: *Mail-Server* für Empfang und Speicherung, *Mail-Client* zum Lesen, Zugriff mit IMAP (mail access protocol)

Bulletin Boards (News, NNTP)

- Network News Transport Protocol, 1986
- Textnachrichten ähnlich wie Email, aber Empfänger ist keine Person, sondern eine *news group*,
- Nachrichten werden archiviert auf *News Servern*, gelesen mit *News Reader*, Archive der Server werden regelmäßig synchronisiert
- diese dezentrale System heißt *usenet*, ist wichtiger Teil der frühen (d.h., nicht kommerziellen) Internet-Kultur, durch Überwachungswirtschaft weitgehend zerstört (EEE: embrace („google groups“), extend, extinguish)
- partielles Archiv
  `https://archive.org/details/usenethistorical`

Das World-Wide Web: HTTP

- HTTP (Hypertext Transfer Protocol) 1996
  `https://tools.ietf.org/html/rfc1945`
- Beispiel: Anfrage (Request)

```
GET /startseite/ HTTP/1.1
Host: www.htwk-leipzig.de
```

- Beispiel: Antwort (Response)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 62381
...
<!DOCTYPE html><html lang="de-DE"><head>
<title>HTWK Leipzig &#x01C0; Startseite </title>
```


Das World-Wide Web: HTML (CSS, JS)

- eine mögliche Form der Antwort-Daten: HTML (Hypertext Markup Language) `https://html.spec.whatwg.org/`
  - HTML: Struktur (DOM, document object model)
    `<ul><li><div .mark>foo</div></li><li>bar</li></ul>`
- enthält oft eingebettet Programmtexte der Sprachen
  - CSS (cascading style sheets): beschreibt das Aussehen
    ```
    li { list-style-type:square; }
    .mark { color:#505b62; }
    ```
  - JS (Javascript): für Programme zur Veränderung des DOM (und seiner Attribute) im Browser und mit Befehlen, die weitere HTTP-Get-Request auslösen und Antworten verarbeiten (DOM erweitern)

Datenübertragungen durch HTTP, Cookies

- der Zweck von HTTP-Get ist, Daten vom Server zum Browser zu übertragen (um Webseite anzuzeigen)
- es gibt auch HTTP-Post vom Browser zum Server, zur Übertragung von Eingaben (z.B. autotool)
- nicht offensichtlich ist, welche Daten der Browser bei Get preisgibt (neben der eigenen IP-Addresse)
  - Eigenschaften des Browsers, des Betriebssystems
- der Server kann eine Textdatei (Cookie) an einen Response anhängen, diese wird der Browser bei weiteren Requests an diesen Server mitschicken
  - offensichtliche Verwendung: Cookie = Sessionkennung
  - versteckt: zur Wiedererkennung des Browsers (zur Herstellung eines genaueren Personenprofils)

Das World-Wide Web: URLs

- einheitliche Ressourcen-Bezeichner (URL)
  `https://www.htwk-leipzig.de/startseite/`
  Protokoll, Host, Pfad, [Parameter]
- dienen zur eindeutigen und stabilen Bezeichnung einer Informationseinheit
- ein HTML-Dokument (ein DOM) kann URLs enthalten, die dadurch beschriebenen/verwiesenen Ressourcen werden *sofort* in die Seite eingefügt (Bilder) oder dienen der *Navigation*
- durch diese Verweise erhält das WWW die Struktur eines (gerichteten) Graphen (dazu Hausaufgabe)

URLs und ihre Feinde

- die Überwachungswirtschaft ist bestrebt, die Funktionen von URLs (Eindeutigkeit, Stabilität) zu unterlaufen
  - wenn man ewig klicken und scrollen muß, um eine Information zu finden, wurde damit die „Interaktivität“ der Seite „verbessert“, d.h. in Wirklichkeit kann man:
  - mehr Werbung anzeigen, mehr Bewegungsdaten erheben (ergibt genaueres Personenprofil)
- und URLs überhaupt zu verschleiern
  - Ressource soll nicht durch den URL gefunden werden, sondern durch die Suchmaschine
  - AMP (accelerated mobile page): ersetzt tatsächlichen URL durch Verweis auf (eigenen!) Zwischenspeicher

Zentralisiert, föderiert

- ein *zentralisierter* Dienst wird durch einen einzigen Dienstanbieter erbracht (Bsp: Zoom, . . .)
- ein *föderiertes* System besteht aus (vielen) Servern unterschiedlicher Betreiber (Bsp: Mail, WWW)
- dazu müssen Protokolle *exakt* und *offen* sein: dann passen die Server zueinander und zu den Clients
- wer Ideen und Ressourcen hat, kann (zu gegebenem Protokoll) andere/bessere Server und Clients programmieren, damit entsteht idealerweise ein Markt für den technischen Fortschritt
- bei *nicht offenem* Protokoll: nur Client-Software des Dienstanbieters verwendbar (dieser wird das ausnutzen)

Bsp. für offene Protokolle, föderierte Dienste

- instant messaging: Protokoll XMPP
  `https://tools.ietf.org/html/rfc6120`,
  Server: Jabber.org, oder selbst betreiben;
  Clients: Gajim, Conversejs, . . .
- real time communication zw. Browsern (z.B. in BBB, Jitsi)
  `https://www.w3.org/TR/webrtc/` (26. Jan. 2021)
- dezentraler Datei-Austausch, z.B. in einem lokalen Netz, zwischen verschiedenen (eigenen) Geräten: PC, Tablet
  `https://syncthing.net/`
  None of your data is ever stored anywhere else other than on your computers. There is no central server that might be compromised, legally or illegally. `https://f-droid.org/en/packages/com.nutomic.syncthingandroid/`

Software-Lizensierung

- es geht dabei um Benutzungsrechte, ist unabhängig von Urheberrechten (diese sind nicht übertragbar)
- Software erscheint in diesen Formen:
  - maschinennah (zur Ausführung),
  - menschennahe (Quelltext, zur Weiterentwicklung)
  - unsichtbar hinter einer Dienst-Schnittstelle
- *freie* Software: Quelltext ist verfügbar, änderbar
- nur freie Software kann objektiv auf Eigenschaften (Sicherheit, Zweckbindung) geprüft werden
- GPL (*strong copyleft*) Änderungen und Ergänzungen am Quelltext stehen unter gleicher Lizenz `http://www.gnu.org/licenses/quick-guide-gplv3.html`

Freie Software, öffentliches und privates Geld

- nichts ist umsonst, auch nicht Software-Herstellung
- Arbeitszeit wird von öffentlichen und privaten Stellen bezahlt (z.B. HTWK: autotool, IBM: Fedora GNU/Linux)
- andere (öfftl.) Stellen sparen dadurch Geld, werden produktiver (private: mehr Umsatz, Steuern, Stellen)
- Investition in freie Software ist sinnvoll (Werbung, Test)
- öffentl. Invest. in unfreie Software (z.B. Opal/Onyx) nicht.
  - Mai 2020 `https://www.opensourcelms.de/`
  - 2017 `https://www.zdnet.com/article/linux-flagship-munichs-u-turn-install-win`

Dienste: lokal und fern, gestern und morgen

- 1960: Stapelbetrieb: Auftrag im Rechenzentrum abgeben
  1970: Time-Sharing: viele Terminals, ein Zentralrechner
  1980: personal computer (jeder seinen eigenen)
- 2000: Web-Clients: Zugriff auf fremde Rechner („Cloud“) vom eigenen Web-Browser aus
  2010: Apps: . . . vom eigenen „smart“-Phone.
- (überwachungs)wirtschaftliche Motivation für Apps:
  - App nur im App-Store, dazu braucht man Account,
  - Datenverarbeitung der App ist schwerer zu kontrollieren (für Browser gibt es Adblocker)
  - Apps halten Benutzer fest (im Browser kann man einfach neuen Tab mit neuem URL aufmachen)

Hausaufgaben

WS21: Aufgaben 1, 4, 5

1. der Graph des WWW: hat wieviele Knoten (wieviele Webseiten gibt es)?

   Vorsicht: zwei Interpretationen von *Webseite* denkbar

   - die einzelne Seite, gegeben durch ihren URL,
   - web site (site im Sinne von Ort, für mehrere Seiten) gegeben durch Domain-Namen des Servers.

   was ist der durchschnittliche Kantengrad (wieviele Links sind auf einer Seite)?

   (recherchieren Sie nach möglichst aktuellen und glaubhaften Schätzungen in wissenschaftlichen

Publikationen, überprüfen Sie deren Plausibilität durch eigene Anschauung)

Wenn jede Webseite 1 kByte Text enthält (ist das realistisch? finden Sie dazu evtl. andere Werte?), wieviele Byte hat das WWW insgesamt?

Wenn man das alles auf Datenträger (micro-SD oder ähnlich) speichern würde (z.B. zum Archivieren): welches Volumen/welche Masse/welche Kosten?

(PS: 1. es ist natürlich schon gespeichert, aber meist nicht langfristig, 2. das meiste lohnt sich wohl nicht zu archivieren, denn es ist Unsinn von und für Maschinen (SEO–search engine optimisation), 3. vergleiche `https://archive.org/`)

2. Erklären und erproben Sie das Firefox-Addon `https://addons.mozilla.org/en-US/firefox/addon/temporary-containers/`

   Legen Sie dazu zunächst ein neues Browser-Profil an (`about:profiles`) und installieren Sie das Addon dort.

   Beobachten Sie dann das Verhalten bei gleichzeitiger Verwendung (Login) von autotool und Opal, mit und ohne Addon.

3. Stellen Sie die Lizenzen der Lernmanagementsysteme Opal und des zugrundeliegenden Systems (open)Olat fest. Was wäre anders, wenn Olat unter GPL stünde?

4. (a) Bestimmen Sie einige URLs, die direkt das Studium

betreffen (z.B. Ihre Studienordnung, Modulbeschreibung, akademischer Kalender) und schätzen Sie deren Langlebigkeit ein.

(b) Bestimmen Sie den URL der Tagesschau vom 11. Januar 2022 in der ARD-Mediathek (oder konkretes anderes Datum, konkrete andere Sendung im öfftl.-rechtlichen Rundfunk) (1. einfach: die Seite mit eingebettetem Player, 2. nicht so einfach: die tatsächliche mp4-Datei)

(c) Vgl. dazu auch `https://arstechnica.com/tech-policy/2021/10/viewing-website-html-code-is-not-illegal-or` Jon Brodkin, 10/25/2021

(d) was ist das Geschäftsmodell von „kostenlosen“ URL-Verkürzungs-Diensten?

5. Um die Silo-Wirkung (das Eingesperrtsein der Benutzten) der monopolistischen sogenannten sozialen Netzwerke etwas zu mildern, hat die EU das *Recht auf Datenportabilität* definiert.

   Geben Sie die Primärquelle an (das Gesetz).

   Wie soll das wirken? Tut es das? (recherchieren Sie dazu Sekundärquellen)

   Sehen Sie Anwendungen auf Verarbeitung Ihrer Daten durch HTWK?

Die Überwachungswirtschaft

Einleitung, Überblick

- an der Oberfläche:
  - Versteigerung von Online-Werbeplatz
  - SEO: Irreführung der (vermuteten) Ranking-Verfahren von Suchmaschinen
  - dark patterns: Irreführung der Webseitenbesucher
- im Hintergrund: Anlegen, Zusammenführen und Verfeinern von Personenprofilen
- Auswirkungen/Gefahren
  - Mißbrauch d. Profildaten durch Dritte (privat und Staat)
  - Lenkung d. öffentl. Diskussion, d. staatlichen Handelns

Suchmaschinen

- bestimmt (ursprünglich) zu gegebener *Anfrage*
  (Menge oder Folge von Wörtern, mit Operatoren)
  als *Antwort* eine Menge von URLs, die auf Ressourcen verweisen, deren Inhalte zur Anfrage passen
  geordnet nach Passung und (angenommener) Qualität
- Suchdienst-Anbieter betreibt *web crawler*, erstellt damit *Index* (Relation zwischen Schlüsselwörtern und URLs)

```
66.249.69.45 - - [04/Feb/2021:11:54:42 +0100]
"GET /robots.txt HTTP/1.1" 200 310 "-"
"Mozilla/5.0 (compatible; Googlebot/2.1;
+http://www.google.com/bot.html)"
```

- in `robots.txt` steht `User-agent: * Disallow: /`

Suchmaschinen-„Optimierung“

- Anbieter der URLs betreiben SEO (search engine optimisation) mit dem Ziel, für bestimmte Begriffe im Index „weit oben“ zu erscheinen
- *Page Rank* (Brin und Page 1998, vgl. `https://arxiv.org/abs/1002.2858`)
  begünstigt Seiten, auf die oft verwiesen wird
  Link Farm: Seiten, die nur dem Verweisen dienen
- Resultat: *SEO spam*. Für Seitenbetreiber gut, für Besucher schlecht, Information fehlt oder ist kopiert
- um die dadurch von der Suche kommenden Besucher
  - als eigene Kunden zu gewinnen
  - an die Anzeigenwirtschaft zu verkaufen

Publizieren und Werben

- (das behandeln Sie alles noch viel ausführlicher in weiteren LV, hier nur ein Überblick sowie im folgenden informationstechnische Aspekte)
- Publikation als reine Information: Bsp: Bund, Land, Stadt: Gesetze, Verordnungen, Verwaltungsakte
- Publikation als reine Werbung für anderes Geschäft: Bsp: Bäcker informiert über Kuchen, Ziel ist Bestellung, Kauf.
- Publikation als eigenes Geschäftsziel,
  Bsp: Autor (Verlag) verkauft Buch, Zeitschrift, Zeitung
- deren Kaufpreis wird mglw. reduziert durch bezahlten Inhalt (Werbung), eventuell sogar auf Null,
- damit wird Werbung zum eigentlichen Geschäftszweck
- Ü: einordnen: Webseite einer Fluglinie, einer Zeitung

Online-Werbung

- klassisch (Druck): Werbekunde kauft Anzeigenplatz in Publikation, Leser kauft Publikation, Leser sieht Anzeige.
- derzeit übliche Online-Werbung:
  - Besucher (Browser) B ruft Webseite W auf
  - Webseite enthält Code, der Browser-Kennung von B an Anzeigen-Händler H übermittelt
  - H bestimmt das Personenprofil P von B,
  - H bestimmt durch Auktion den Anzeigenkunden K, der für Merkmale von P den höchsten Preis bietet,
  - H liefert Anzeige von K an B aus,
    zieht Geld von K ein, zahlt Anteil davon an W

  Anzeigenhändler überwacht Leser, behält dessen Daten

Versteigerung von Suchbegriffen

- Hal Varian: *Position Auctions*, J. Industry Org. 2007,
  `https://people.ischool.berkeley.edu/~hal/Papers/2006/position.pdf`

  When a user's search query matches a keyword, a set of ads is displayed. These ads are ranked by bids . . . and the ad with the highest bid receives the best position; i.e., the position that is mostly likely to be clicked on . . . . If the user clicks on an ad, the advertiser is charged . . . the bid of the advertiser below it in the ranking.
- ist verallgemeinerte Zweitpreis-Auktion, die optimale Strategie dafür ist, den wahren Wert zu bieten
  bei Erstpreis-Auktion nicht, dort wird man unterbieten, weil man sonst den Abstand zum zweiten verschenkt

Eindeutige Personenkennung

- Anzeigenhändler erzielt höchsten Gewinn, wenn die einzelne Browser-Aktion (HTTP-Request) anderen (früheren) Aktionen derselben Person zugeordnet werden kann und für diese Person bereits Merkmale bestimmt und gespeichert wurden
- anstatt die Zuordnung bei jedem Request zu raten, kann man sie in einem Cookie abspeichern:
  B ruft W_1, dieser verwendet H, H sendet Cookie an B.
  B ruft W_2, dieser verwendet auch H, H fragt B nach Cookie und erkennt ihn dadurch wieder
- kann man (im Browser) unterbinden durch:
  - alle Cookies (und JS) von Dritten ablehnen
  - Cookies *isolieren*: getrennte Container C_i für W_i

Sogenannte soziale Netzwerke

- tatsächliche soziale Beziehungen modelliert als Graph
  (jeder Mensch ist ein Knoten,
  jede Beziehung/Interaktion eine Kante)
  Ü: dieser Graph hat kleinen Durchmesser
- benachbarte Knoten haben ähnliche Profile
  (das ist der praktische Nutzen beim Anzeigenverkauf)
- die sogenannten sozialen Netzwerke verwerten den Teilgraphen ihrere Mitglieder. . .
- mit Schattenprofilen für Nicht-Mitglieder (Personen auf Fotos, in Kontakdaten, . . .) dadurch genauere Profile

Auswirkungen massiver Datensammlung

- Werbung ist noch nicht das Ende der Welt?
- für Anzeigen-Auktionen angelegte Personenprprofile werden *beliebig lange* zu *beliebigen Zwecken* verwertet
- stellen Angriffsziel dar (Spionage durch Konkurrenten) (z.B. Email-Dienstleister gegen Online-Versandhändler)
- ist Ziel staatlicher Überwachung, das ist gefährlich bei schwacher parlamentarischer und richterlicher Kontrolle
- werden verkauft an Dritte (vgl. `https://ig.ft.com/how-much-is-your-personal-data-worth/`, Financial Times 2017)
  die damit wirtschaftliche und politische Ziele verfolgen

Informationsfilterung, Microtargeting

- „hab ich im Internet gelesen“ bedeutet oft: „wurde mir von Suchmaschine (oder anderer Software) vorgeschlagen“
- das Ziel der Betreiber ist aber nicht die objektive Information, sondern die Gewinnmaximierung
- Vorschläge werden personenbezogen so ausgewählt (gefiltert), daß bisherige Meinungen bestätigt werden
- ein Akteur kann in Filterblase F_1 die Meinung M_1 aufstellen und in (disjunkter) F_2 die (gegenteilige) M_2, in Öffentlichkeit würde der Widerspruch auffallen
  `https://www.bpb.de/gesellschaft/digitales/digitale-desinformation/290522/microtargeting-und-manipulation-von-cambridge-analy`

Überwachung und Bürgerrechte

- Gesetzgeber (Land Hessen 1970, Bund 1978, EU 2018) definiert *Recht auf informationelle Selbstbestimmung* („Datenschutz“, aber geschützt wird der Bürger)
- zunächst als Abwehrrecht gegen den Staat (Anlaß war automatisierte Auswertung einer Volkszählung)
- zwischen Privaten (Bürger, Wirtschaft) besteht Vertragsfreiheit—aber beiden Parteien muß der Vertragsgegenstand tatsächlich klar sein, hier: Umfang und Zweck der Verarbeitung personenbezogener und -beziehbarer Daten
- Überwachungswirtschaft verschleiert und verharmlost
- Gesetzgeber (EU) definiert Verbraucher-Rechte, möchte aber auch einheimische Überwachungswirtschaft fördern

Irreführende Benutzerschnittstellen

- verführen Webseiten/App-Benutzer zu nicht beabsichtigen Handlungen, (Verlust v. Geld, Daten, Zeit)
- *dark pattern* Harry Brignull 2010, `https://darkpatterns.org/types-of-dark-pattern.html`
- Norwegischer Verbraucherverein 2018 `https://fil.forbrukerradet.no/wp-content/uploads/2018/06/2018-06-27-deceived-by-design-final.pdf`
- zur Unterhaltung: hier geht es vorwiegend um Verlust von Zeit und Nerven: `https://userinyerface.com/`
  A *worst-practice* UI experiment (Bagaar.Be, 2019?)
  (Lösungshinweise: `https://news.ycombinator.com/item?id=20344565`)

Aktuelle Informationen und Gesetzgebungsvorhaben

- Der Sächsische Datenschutzbeauftragte: Tätigkeitsbericht 2019 `https://www.saechsdsb.de/taetigkeitsberichte`
- Europäische Kommission: (Vorschlag) *Gesetz über digitale Dienste (DSA)* (15. 12. 20) `https://eur-lex.europa.eu/legal-content/DE/TXT/HTML/?uri=CELEX:52020PC0825`
- Europäische Kommission: (Vorschlag) *Gesetz über digitale Märkte (DMA)* (15. 12. 20) `https://eur-lex.europa.eu/legal-content/`

`DE/TXT/HTML/?uri=CELEX:52020PC0842`

Hausaufgaben

1. Fassen Sie wesentliche Aspekte von `https://atg.sd.gov/docs/20201216COMPLAINT_REDACTED.pdf` zusammen. Benutzen Sie auch (aber nicht nur) Sekundärquellen.
2. Fassen Sie den aktuellen Konflikt zwischen Facebook und Apple zusammen, siehe z.B. NZZ vom 31. 1. 21 `https://www.nzz.ch/technologie/facebook-und-apple-der-heranziehende-datenk1599202`
3. (Aufgabe von voriger Woche) Recht auf Datenportabilität

4. zu soziale Netzwerken:

   Was ist eine *Kevin-Bacon-Zahl*? Bestimmen Sie diese für den Darsteller der Titelfigur in *Ein Fall für Rettig* `https://www.htwk-leipzig.de/no_cache/de/hochschule/aktuelles/newsdetail/artikel/944/`

   Definieren Sie den Begriff *Erdös-Zahl*, bestimmen Sie meine. (Bonus: und die von Kevin Bacon.)

   Schätzen Sie für den jeweiligen Graphen (Schauspieler, Mathematiker): Anzahl der Knoten, Knotengrad, Durchmesser.

Ausblick, Wiederholung

Definition, Ziele

- Informatik =
  Wissenschaft von (der Verarbeitung symbolisch repräsentierter Information durch) Algorithmen
  als Ingenieurdisziplin:
  Entwurf, Konstruktion, Betrieb von Softwaresystemen
- Ziele der VL für Studenten im Nebenfach:
  - effiziente Kommunikation mit Informatik-Fachleuten
  - Methoden zu Beschreibung technischer Systeme
  - Allgemeinbildung, insbesond. zur privatwirtschaftlichen und staatlichen Überwachung mittels IT-Systemen

Themen

- Hardware . . . und Informationsdarstellung
  Wahrheitswerte, logische Schaltungen
  Maschinenzahlen
  Medien (Text, Bild, Ton, Film)
- Software . . . Informationsverarbeitung
  Namen, Argumente, Typen, Iteration
- Netze
  Codes (Redundanz), Kompression
  Standards, Protokolle, Dienste,
  wirtschaftliche und gesellschaftliche Aspekte

Struktur-Modell: Baum, Graph

- Bäume (Terme) zur Beschreibung hierarchischer Struktur
- Graphen zur Beschreibung von Netzen
- Wiederholung/Übung: Code mit Bitbreite 6 und Hamming-Abstand 3
- Wiederholung/Übung: Erdos-Zahl, Abschätzungen

Programmierung?

- wir hatten Geradeaus-Programme, dann vorsichtig erweitert
- „richtiges“ Programmieren: brauchen Sie nicht, das sollen die Fachleute machen. Die fangen aber auch so an!
- was Sie dabei gelernt haben, und beim Umgang mit jedem technischen System brauchen:
  - Trennung von *Syntax* (was man sieht/schreibt/bedient) und *Semantik* (was es bedeutet),
  - Trennung von *Spezifikation* (was es tun soll) und *Implementierung* (wie es das tut)

Informatik und Medien (Beispiele)

- z.B. Grafik: Darstellung, Speicherung, Kompression, Übertragung, Kodierung
- algebraische (baum-artige) Beschreibung,
  vgl. J.W.: *Eine einfache Grafik-Programmiersprache für die Einführung in die Informatik*, 2021 `https://www.imn.htwk-leipzig.de/~waldmann/talk/21/abp/`
- Ausblick: algebraische Beschreibung von Musik
  Bsp: Operatoren `stack` (gleichzeitig) und `cat` (abwechselnd) für periodische Muster in Tidalcycles (A. McLean) `https://www.imn.htwk-leipzig.de/~waldmann/edu/ws21/cm/folien/#(127)`

Prüfung

- Zulassung: Hausaufgaben autotool
- Teil-Leistung: Hausaufgaben-Präsentation
- Teil-Leistung: elektronische Hausarbeit (Prüfungsform PH-D)
  konkrete Durchführung: interaktiv mit autotool.
- Prüfungsvorbereitung:
  - Nacharbeiten der Hausaufgaben.
    (aus Semester, aus Testprüfung; Sie können Aufgaben-Instanzen austauschen, ich stelle weitere)
  - Diskussion im Forum des Opal-Kurses.