SS 16

Luke Hain

26. April 2016

Inhaltsverzeichnis

I Computer Networks **5**

1 Vorlesung **6**
- 1.1 Einführung . . . 6
- 1.2 Bitübertragungsschicht . . . 7
  - 1.2.1 Nachrichtentechnische Kanäle . . . 7
  - 1.2.2 Übertragungsmedien . . . 8
  - 1.2.3 Mehrfachnutzung von Kanälen . . . 10
  - 1.2.4 Datenübertragung . . . 11
  - 1.2.5 Beispieltechnologien . . . 12
  - 1.2.6 Digitaler Netzzugang über Kabelmodem . . . 12
- 1.3 Netztechnologie 1 . . . 12
  - 1.3.1 Medienzugriff . . . 12
  - 1.3.2 Ethernet . . . 13
  - 1.3.3 Switches in der Sicherungsschicht . . . 15
  - 1.3.4 Drahtlose Netze für PAN und LAN (3.26) . . . 17

2 Übung **19**
- 2.1 Einführung . . . 19
  - 2.1.1 . . . 19
  - 2.1.2 . . . 20
  - 2.1.3 . . . 20
  - 2.1.4 . . . 21
- 2.2 Bitübertragungsschicht . . . 22
  - 2.2.1 Nyquist-Theorem . . . 22
  - 2.2.2 Pulsecodemodulation . . . 23
  - 2.2.3 Modulation . . . 23
  - 2.2.4 Leitungskodierung . . . 24
  - 2.2.5 Multiplex . . . 24
- 2.3 Netztechnologien 1 . . . 25
  - 2.3.1 Ethernet . . . 25
  - 2.3.2 Switches . . . 26
  - 2.3.3 Transparent Bridges . . . 27
  - 2.3.4 802.11 WLAN . . . 27

II Theoretical Informatic and Logic **29**

3 Vorlesung **30**
3.1 Prädikatenlogik erster Stufe . . . 30
3.2 Prädikatenlogik erster Stufe . . . 31
3.3 Syntax/Substitutionen . . . 32
3.3.1 Komposition von Substitutionen . . . 32
3.3.2 Beschränkung von Substitutionen . . . 32
3.3.3 Anwendung von Substitutionen auf Formeln . . . 33
3.3.4 Substitutionen und Formeln . . . 33
3.3.5 Satz 4.18 . . . 33
3.3.6 Beweis Hilfsaussage aus Satz 4.18 . . . 35
3.3.7 Varianten . . . 35
3.4 Semantik . . . 35
3.4.1 Relationen und Funktionen . . . 35
3.4.2 Interpretationen . . . 36
3.4.3 Herbrand-Interpretationen . . . 37
3.5 Modelle . . . 38
3.5.1 Modelle für abgeschlossene Formeln . . . 38
3.5.2 Modelle für nicht-abgeschlossene Formeln . . . 38
3.6 Äquivalenz und Normalform . . . 39
3.6.1 Semantische Äquivalenz . . . 39
3.6.2 Pränexnormalform . . . 40
3.6.3 Skolem-Normalform . . . 41
3.6.4 Klauselform . . . 42

4 Übung **43**
4.1 Prädikatenlogik - Syntax . . . 43
4.1.1 Konstruktion von Teiltermen . . . 43
4.1.2 Über Nachbarn . . . 44
4.2 Substitutionen . . . 45
4.2.1 Substitutionskomposition ist eine Substitution . . . 45
4.2.2 Eingenschaften von Substitutionen . . . 45

III Computer Architecture **47**

5 Vorlesung **48**
5.1 Einführung . . . 48
5.1.1 Big Data . . . 48
5.2 Vorlesung . . . 48
5.2.1 ZIH . . . 48
5.2.2 Begriffe und Definitionen . . . 48

5.3 VL . . . 49
5.3.1 Modifiziertes Dreiphasenmodell zum Entwurf eines RS . . . 49
5.3.2 Architektur-Definition (Tanenbaum) . . . 49
5.3.3 Architektur-Definition (Hennessy/Patterson) . . . 49
5.3.4 Einflusskomplexe . . . 50
5.3.5 Entwurf eines Rechnersystems . . . 51
5.3.6 Architectural Trends . . . 52
5.3.7 Bemerkungen zum klassischen Digitalrechner . . . 53
5.3.8 Aufgaben und Ziele der Rechnerarchitektur . . . 54
5.3.9 Klassifizierung nach Flynn . . . 54
5.4 Intel Prozessoren . . . 55
5.4.1 Ursprung . . . 55
5.4.2 Intel 80386 . . . 55
5.4.3 Intel 80486 . . . 56
5.4.4 Pentium . . . 57
5.4.5 Pentium MMX . . . 57
5.4.6 Pentium Pro . . . 58
5.4.7 Pentium 2 / Pentium 3 . . . 59
5.4.8 Pentium 4 . . . 59
5.4.9 Pentium M/ Core Solo/ Core Duo . . . 60
5.4.10 Intel64 Prozessoren (Auswahl) . . . 61
5.4.11 Multicore Prozessoren . . . 61
5.4.12 Intel Core Microarchitektur Core2Duo . . . 62
5.4.13 Intel Nehalem Mikroarchitektur . . . 62

6 Übung **63**
6.1 Einführung . . . 63
6.1.1 von-Neumann . . . 63
6.1.2 v.Neumann vs. Harvard . . . 64
6.1.3 Def. von Brooks vs Giloi . . . 64
6.1.4 RA-Definition Begriffe . . . 65
6.2 Einführung . . . 66
6.2.1 Moores Law . . . 66
6.2.2 Klassifikationen nach Flynn . . . 68

IV Database **69**

7 Vorlesung **70**
7.1 Einführung . . . 70
7.2 Konzeptueller Entwurf . . . 71
7.2.1 Drei Phasen des Datenbank-Entwurfs (4, ff.) . . . 71
7.2.2 Lebenszyklus einer Datenbank . . . 71

7.2.3 Prinzip eines Datenmodells (16) . . . 71
7.2.4 Entity-Relationship-Modell . . . 72
7.3 Konzeptueller Entwurf . . . 73
7.3.1 Grundlagen (5) . . . 73
7.3.2 Primärschlüssel (8) . . . 74
7.3.3 ER-Modell Relationales Modell . . . 74
7.4 Relationale Algebra . . . 75
7.4.1 Motivation . . . 75
7.4.2 Relationale Algebra . . . 76
7.4.3 Basisoperationen . . . 76
7.4.4 Abgeleitete Operationen . . . 77

8 Übung **79**
8.1 Einführung . . . 79
8.1.1 . . . 79
8.1.2 . . . 79
8.1.3 . . . 80
8.1.4 . . . 80
8.1.5 . . . 80
8.2 ER-Modellierung . . . 80
8.2.1 Prof-Stud . . . 80
8.2.2 ER-Bahn . . . 80
8.2.3 ER-Bsp . . . 82
8.2.4 Vererbungshierarchie - Relationsschema . . . 82

V Hardware Laboratory **84**

VI C++4CG **85**

Teil I

Computer Networks

Kapitel 1

Vorlesung

1.1 Einführung

- Anwendungsfelder Rechnernetze (1.4)
  - Geschäftsanwendungen - gemeinsame Nutzung von Resourcen
  - Privatbereich - Informationszugriff (z.B. WWW, IM)
  - Mobile Benutzer - Textnachrichten, ...
  - Gesellschaftliche Aspekte - Copyright, Profile, ...
- Client Server Modell (1.5)
- Peer-to-Peer Communication (1.6)
- Basis-Netzstruktur (1.7)
  - Übertragungsmodi
    - Verbindungsorientiert
    - Verbindungslos (z.B. IP)
    - Leitungsvermittelt
    - Paketvermittelt (flexibler, ressourcenschonend)
- Schichtenarchitektur - ISO/OSI Referenzmodell (1.8)
  - International Organization for Standardization
  - Open Systems Interconnection
  - Schichtenübersicht auf 1.8 ff.
- Integriertes Referenzmodell (Tanenbaum) (1.11)
  - Protokollimplementierung oft abweichend vom Referenzmodell

- Besipiel Datenübertragung (1.12)
- Schichteneffizienz (1.13)
- Dienste - Begriffsklärung (1.14)
  - Beispiel Ablaufdiagramm (1.15)
- Netzkopplung - Basis-Topologien
  - Punkt-zu-Punkt-Kanäle (Unicast)
  - Rundsendekanäle (Broadcast)
  - Klassifizierung nach Ausdehnung (1.17)
    - Pan - Personal Area Network
    - LAN - Local Area Network
    - MAN - Metropolitan Aria Network
    - WAN - Wide Area Network (1.18)
  - Mobilität || Leistung (1.19)
  - Konzepte - Layer-N-Gateway(1.20)
  - Beispiel (1.21)
- Internet(1.22 ff)
  - Internet
    - Geschichte des Internet (1.24 ff)
    - Normen (1.26)
  - Intranet (1.22)

1.2 Bitübertragungsschicht

1.2.1 Nachrichtentechnische Kanäle

- Aufgabe: Physikalische Bitübertragung mittels Transformation in elektromagnetisches Signal
- Daten $\rightarrow$ Kanal $\rightsquigarrow$ Störeinflüsse $\rightarrow$ Daten

Kenngrößen (2.4 ff)

- Bandbreite B: Breite des Frequenzbereichs eines Kanals, in dem ohne größere Dämpfung übertragen wird
- Baudrate
- Bitrate
- Nyquist Theorrem $b < 2 \cdot B \cdot ld(S)$
  - Erweiterung durch Shannon $b < B \cdot ld(1 + SNR)$
  - Kombination $b < \min(2 \cdot B \cdot ld(S)\ ;\ B \cdot ld(1 + SNR))$

Leitungscodes

- Wie soll Folge von 0en und 1en übertragen werden?
- NRZ: **N**on-**R**eturn-to-**Z**ero (2.6)
- Manchester-Codierung
- NRZI: NRZ-**I**nverted (2.7)
  - Signaländerung bei 1, keine Signaländerung bei 0
  - Vortei: hohe Netto-Datenrate
  - Nachteil: Probleme bei langer Folge von Nullen
  - Lösung: 4B/5B Code
    - jeweils 4 Bits Daten werden auf 5-Bit-Muster abgebildet $\rightarrow$ 25
    - durch 4B/5B-Code treten niemals mehr als 3 Nullen nacheinander auf

1.2.2 Übertragungsmedien

Elektrische Leitungen

- Twisted Pair (2.8)
  - isolierte Kupferdräthe von 0,4 bis 1mm Stärke
  - Paarweise verdrillt $\rightarrow$ Reduzierung von Störungen
  - Üblicherweise 4 Paare pro Kabel
  - Mehrere Kilometer Reichweite, mehrere MBit/s, preiswert
  - Signal aus Spannungsdifferenz zwischen den 2 Kabeln übertragen
  - Cat 3
  - Cat 5
  - Cat 6

  - Cat 7

- Koaxialkabel (2.9)
  - mehrere km, mehrere MBit/s, T-stecker ode rTap
  - 50-Ohm-Kabel: für digitale Übertragung
  - 75-Ohm-Kabel: für analoge Übertragung und Kabelfernsehen
  - Kabelfernsehen → Breitband-Koaxialkabel, häufig mit analoger Übertragung bis ca. 1 GHz, bidirektionaler Ausbau für Internet-Zugang via Kabel

Optische Leitungen und Sichtverbindung

- Optische Leitungen
  - Lichtwellenleiter (LWL) / "Glasfaser"
    * bis TBit/s-Bereich, über viele km Entfernung
    * Monomodefaser: nur eine ausbreitungsfähige Wellenform
    * Multimodefaser: verschiedene ausbreitungsfähige Wellenformen
    * Gradientenfaser: schrittweise Änderung des Brechungsindex

Sichtverbindung

- Infrarotverbindung
- Richtfunkstrecken

Satelliten / Zellularfunk (2.11)

- Satelliten
  - Getrennte Aufwärts-/Abwärtsbänder
  - Bandbreite von 500MHz, z.B. in mehrere 50 MBit/s - Kanäle oder 800 digitale Sprachkanäle mit 64 kBit/s
  - Zuordnung kurzer Zeitabschnitte zu einzelnen Kanälen (Zeitmultiplex)
  - Lange Laufzeiten (ca. 250 bis 300ms)
- Zellularfunk
  - Aufteilung eines geographischen Bereichs in Funkzellen mit spezifischen Frequenzbändern
  - Beispiel: GSM (Global System for Mobile Communication)

Strukturierte Verkabelung (2.12

- Ziel: Systematische, gut wartbare und erweiterbare Kabelinfrastruktur
- Trennung in drei wesentliche Bereiche (jeweils sternförmig hierarchisch)
  - Primärebene
  - Sekundärebene
  - Tertiärebene

1.2.3 Mehrfachnutzung von Kanälen

Frequenzmultiplex (2.13)

- getrennte Frequenzbänder (mit z.B. 3000 Hz) und zwischengeschaltete Sperrbänder (mit z.B. 500 Hz)

Orthogonales Frequenzmultiplex (Orthogonal FDM, OFDM)

- Überlaguerung der Kanäle ohne Sperrbänder → effizienter
- Empfänger: Trennung der Signale mehrerer Bänder durch schnelle Fouriertransformation
- Einsatz: Wlan, Kabelnetze, 4G Mobilfunk, LTE, ...

Zeitmultiplex (2.14)

- Zyklische Kanalzuteilung

Statistisches Zeitmultiplex

- flexible Zuteilung nach Bedarf

Codemultiplex (CDM, 2.15)

- Didizierte (Kodierungs-)Codes pro Teilnehmerpaar

Wellenlängenmultiplex (WDM)

- Variation von Frequenzmultiplex, indem direkte optische Einkopplung mehrerer Lichtwellenleiter (mit Licht unterschiedlicher Wellenlängen) in einen besonders leistungsfähigen Lichtwellenleiter erfolgt
- entsprechende Wiederauskopplung im Zielsystem

1.2.4 Datenübertragung

Signalklassen (2.16)

- Wert/Zeit kontinuierlich $\leftrightarrow$ Wert/Zeit diskret
- Beispiele (2.17)
  - Wert- und zeitkontinuierlich: analoges Telefon
  - Wertkontinuierlich, zeitdiskret: Prozesssteuerung mit periodischen Messpunkten
  - Wertdiskret, zeitkontinuierlich: digitale Temperaturanzeige
  - Wert- und zeitdiskret: digitale Übertragung mit isochronem Taktmuster; z.B. Sprachübertragung über digitale Kanäle

Beispiel: Telefonsystem (2.18)

Sprachübertragung über digitale Kanäle (2.19)

- Analoge Eingangssignale (Sprache) vor Übertragung im Kernnetz zu digitalisieren: Codec (Coder-Decoder)
- Basis: Abtasttheorem nach Shannon $f_(A) > 2 \cdot f_(G)$
- PCM: Pulse Code Modulation
  - Bsp.: Grenzfrequenz (Telefon) : 3400 z; Abtastfrequenz: 8000 Hz
  - logarithmische Quantisierungsintervalle $\rightarrow$ Quantisierungsfehler begrenzen

Datenübertragung über analoge Kanäle

- Modem: Übertragung digitaler Signale über analoge (2.20) Telefonverbindung
  - Problem: Nicht direkt möglich wegen kapazitiver und induktiver Einflüsse
- Amplitudenabtastung
- Periodenabtastung
- Phasenabtastung
  - Ziel: Deutlich höhere Übertragungsleistung durch gleichzeitige Anwendung mehrerer Modulationsverfahren (2.21)
  - Beispiele
    - QPSK
    - QAM 16
    - QAM 64

1.2.5 Beispieltechnologien

Digital Subscriber Line (DSL, 2.22)

- digitaler Netzzugang über herkömmliche Telefonleitungen
- Datenübertragung und Telefondienst gleichzeitig nutzbar
- Realisierung durch Nutzung höherer Frequenzbereiche
- hohe Datenraten, meist asymmetrisch (ADSL) bzgl. Up-/Downlink
- weitere Varianten:
  - VDSL (Very High Bitrate) : nur über kurze Entfernungen
  - SDSL (Symmetric): GLEICHE dATENRATE AUF Up-/Downlink
- Signaltrennung (Telefon/Daten) und Modulation (basierend auf QAM, 2.23)
  - CAS (Carrierless Amplitude / Pase System)
  - DMT (Discrete Multitone)

1.2.6 Digitaler Netzzugang über Kabelmodem

- Signaltrennung zwischen Kabelfernsehen und Daten:
  - Umwidmung einzelner TV-Kanäle in Datenkanäle
  - Rückkanalfähige Verstärker erforderlich
  - Datenraten theoretisch bis ca. 36 MBit/s, aber SShared Medium", d.h. abhängig von der Zahl der Teilnehmer geringere Datenrate

1.3 Netztechnologie 1

1.3.1 Medienzugriff

ALOHA Protokoll

- historisches Paketfunknetz, University of Hawaii, seit 1979
- dezentrale Stationen, Kommunikation über Zentrale
- unkoordiniertes Wettbewerbsverfahren (stochastisch)
- Kollision auf f_1 bei Zentrale, da Senden stets möglich
- Fehlerbehandlung durch Wiederholung, falls nach Zeit t keine Quittung auf f_2
- kein Mithören während des Sendevorgangs

ALOHA Beispiel

- Pure ALOHA: Max. etwa 18 Prozent des Kanaldurchsatzes
- Slotted ALOHA: Max. etwa 36 Prozent des Kanaldurchsatzes (3.6)

CSMA-Verfahren

- kein Funk, sondern Coaxialkabel
- Abhören vor Senden (CSMA - Carrier Sense Multiple Acces)
- Trotzdem Kollision möglich: (1-persistent CSMA, immer sendebereit)
- nonpersistent CSMA: belegter Kanal wird nicht sofort erneut abgehört, erst nach zufällig verteiltem Zeitintervall; dadurch geringere Kollisionswahrscheinlichkeit
- p-persistent CSMA (slotted): Prüfe Kanal, sende mit Wahrscheinlichkeit p, warte sonst 1 Slot und prüfe wieder

Bewertung der Verfahren (3.8)

CSMA mit "Collision Detection"(CD)

- Mithören während des Sendevorgangs
- Kollisionserkennung dadurch schneller möglich (ohne Warten auf Quittung)
- Funktioniert für ein gemeinsam genutztes Kommunikationsmedium ... (z.B. gemeinsames Kabel bei IEEE 802.3, Lukf bei IEEE 802.11, etc.)
- ... mit mindestens einer Station (Kollisionen mit sich selbst können erkannt werden, z.B. durch Signalreflexion am offenen Kabelende

CSMA/CD-Verfahren Beispiel (3.10 ff.)

1.3.2 Ethernet

IEEE 802.3

- Zugriffsverfahren: 1-persistent CSMA/CD, in Hardware auf Ethernet-Karte realisiert
- Datenrate der Basistechnologie: 10 MBit/s
- Segmentlänge: 500m
- Kabel der Kategorie 5 oder höher bzw. Lichtwellenleiter (dann auch deutlich größere räumliche Ausdehnungen möglich)

- heute grundsätzlich mit Switches und Duplex-Betrieb im Einsatz
- dennoch Kollisionsbehandlung generell mit eingebaut:
  - warte s Slots nach Kollision, s zwischen 0 und $2^n - 1$ bei n vorherigen Kollisionen zufällig gewählt

Rahmenstruktur (3.14)

- Präambel erlaubt Synchronisation mit Empfänger
- EtherType/Size
  - $\leq 1500 \rightarrow$ Länge des Datenfelds
  - $\geq 1536 \rightarrow$ Typ der Daten (z.B. IP, IPv6, etc.), Länge der Daten nicht spezifiziert $\rightarrow$ Interframe Gap als Begrenzer
- Pad zum Auffüllen auf minimale Rahmenlänge wegen Kollisionsverfahren
- Prüfsumme: CRC (Cyclic Redundancy Check), ohne Präambel und SFD

Fast- / Gigabit Ethernet

- Fast Ethernet
  - 1995 als IEEE 802.3u standardisiert
  - Datenrate 100 MBit/s
  - Segmentlänge: 100m bei Kupferkabel, 2km bei Lichtwellenleiter
  - Kompatibilität zu Ethernet und Cat-3-Kabel, noch CSMA/CD unterstützt aber keine Multidrop-Kabel mehr
- Gigabit Ethernet
  - 1999 als IEEE 802.3ab standardisiert
  - Datenrate 1 GBit/s
  - Vollduplex (Standard): kein CSMA/CD mehr $\rightarrow$ keine Beschränkung der Kabellänge
  - Halbduplex: Layer-1-Kopplung über Hub; CSMA/CD mit Modifikationen:
    - Padding - Rahmen immer auf 512 Byte auffüllen
    - Frame-Bursting - mehrere Rahmen in einem Ethernet-Frame übertragen
- 10 / 100 GBit/s Ethernet
  - für optische Verbindungen in WANs $\rightarrow$ siehe Kapitel 4

Ethernet-Varianten für LAN (3.16)

Switched Ethernet: Beispiel (3.17)

- parallele Vermittlung aller Verkehrsströme durch Switch-Hardware
- Vorteil: Keine Kollisionen, jeder Station steht die volle Ethernet-Datenrate zur Verfügung ⇒ Ethernet wird vom SShared Mediumßum SSwitched MEdium"
- Aufteilung der Stationen and einem oder mehreren Switches in unterschiedliche virtuelle lokale Netze (VLAN) möglich ⇒ Sicherheitszonen

1.3.3 Switches in der Sicherungsschicht

Ziele (3.18):

- parallele Vermittlung durch Switches, sequentiell duch Bridges (veraltet)
- Trennung organisatorischer Bereiche/verschieden Verkehrsströme
- Zuverlässigkeit und Sicherheit (gegen Störsignale und unberechtigte Weiterleitung)
- Begrenzung der Netzlast durch selektives Weiterleiten von Nachrichten

Modell (3.19)

Transparent Bridges / Switches (3.20)

- Selbstlernend: Automatischer Aufbau von Routing-Tabellen
- Topologie-Erkennung durch Quelladressen, schrittweiser Tabellenaufbau
- Fluten, falls Zielrechner noch unbekannt
- Löschen von Einträgen nach bestimmter Zeit zur Anpassung an Topologieänderungen

Spanning Tree (3.21)

- Problem: Mehrfachwege → Endlosschleifen
- Lösung: Aufbau eines "überspannenden Baumes"mit eindeutigen Wegen durch dezentralen Algorithmus / kürzester Weg zur Wurzel

Interne Realisierung (3.22)

- Parallele Vermittlung mehrerer Eingangs- an mehrere Ausgangsports
- Hohe Leistung, unterstützt durch Hardware-Realisierung
- Store-and-Forward-Switch: Gesamtes Frame wird im Switch zwischengespeichert, die Prüfsumme wird kontrolliert und erst dann wird weitergeleitet ⇒ einfach; Pufferung und Datenratenanpassung
- Cut-Through-Switch: Andommende Frames werden nach Prüfung der Zieladresse sofort weitergeleitet ⇒ effizienter, kürzere Verzögerung, aber problematisch bei unterschiedlichen Datenraten und bei Fehlern

VLAN - Virtual Local Area Network (2.23)

- Motivation:
  - Flexibilität: Änderung der Zuordnung von Geräten zu lokalen Netzen ohne neue Verkabelung
  - Sicherheits- und Performance-Aspekte

Port-basiertes VLAN (3.24)

- Jeder Port eines Switches wird einem VLAN zugeordnet
- Ports können nur Mitglied eines VLANs sein
- redundante Links zwischen Switches benötigt

Tag-basiertes VLAN - IEEE 802.1Q (3.25)

- Transport mehrerer VLAN-Pakete über einen Link → Tagging der Pakete
- IEEE 802.1Q - Ergänzung des Ethernet-Headers - VLAN-Tag
- letzter VLAN-Fähiger Switch entfernt das VLAN-Tag wieder → Kompatibilität
- VLAN Identifier = 12 Bit
- andere Felder (Priority und CFI) nicht für VLAN genutzt

1.3.4 Drahtlose Netze für PAN und LAN (3.26)

WLAN: IEEE 802.11 (3.27)

802.11 - Medienzugriff mit CSMA/CA (3.28) ff.

- RTS/CTS - Request to Send / Clear to Send
- Hidden terminal: A kann C wegen begrenzter Funkreichweite nicht hören
  - A sendet RTS-Signal an B, und B sendet dann CTS
  - Alle anderen möglichen Sender (C) erhalten das CTS-Signal und stellen ihren Sendevorgang zurück
- Exposed terminal (unnötiges Warten, hier durch B bei Senden nach links)
  - C sendet RTS an möglichen anderen Empfänger
  - Falls dieser beriet, erhält C das CTS und kann übertragen (unabhängig von B)

Bluetooth

- drahtlose Ad-Hoc-Piconetze (<10m), billige Ein-Chip-Lösung
- offener Standard: IEEE 802.15.1
- Einsatzgebiete:
  - Verbindung von Perpheriegeräten
  - Unterstützung von Ad-Hoc-Netzen
  - Verbindung verschiedener Netze (z.B. drahtloses Headset mit GSM)
- Frequenzband im 2,4 GHz- Bereich; Integrierte Sicherheitsverfahren (128-Bit-Verschlüsselung)
- Datenraten:
  - 433,9 kBit/s asynchronous-symmetrical
  - 723,2 kBit/s / 57,6 KBit/s asynchronous-asymmetrical
  - 64 kBit/s synchronous, voice service
  - Erweiterungen bis zu 20 Mbit/s (IEEE 802.15.3a: UWB (Ultra Wide Band)

ZigBee (3.30

RFID - Radio Frequency Identification (3.31)

- Klasse-1-Tags:
  - bestehen aus Antenne und RFID-Chip
  - 96-Bit-Identifikator, kleiner Speicher, passiv
  - geringer Preis, lässt sich z.B: auf Produkte aufkleben
- Lesegerät:
  - aktiv, leistungsfähig, MAC-Protokolle
  - sendet Trägersignal, wird von Tag reflektiert
- Backscatter: Tag überlagert das Trägersignal mit eigenen zu sendenden Bits → Lesegerät filtert dies Bits aus
- Mehrfachzugriff: modifizierte Version von Slotted ALOHA

NFC - Near Field Communication

- kontaktloser Datenaustausch über Kürzeste Distanzen (4cm)
  - Auflegen/anlegen des Transmitters an Lesegerät erforderlich
- Datenübertragunsrate bis zu 424 kBit/s
- Übertragung
  - verbindungslos: passive RFID-Tags
  - verbindungsorientiert: aktive Transmitter (z.B. Smartphone)
- mögliche Anwendungen
  - Bezahlung per Smartphone oder Smartcard
  - Smartphone als Türschlüssel
- Kritik
  - Distanz als Sicherheitsfeature ungeeignet (durch große Antennen bis zu 1m möglich)
  - NFC-Sicherheitsmechanismen unzureichend

Kapitel 2

Übung

2.1 Einführung

timo.schick@tu-dresden.de

2.1.1

a) Sterntopologie: Ein zentrales Element(Sternkoppler), jeder Rechner benötigt eine Leitung zu Sternkoppler $\rightarrow 5$

b) Jeder mit Jedem $= 4 + 3 + 2 + 1 = 10$

c) (1) $l(n) = n$ bei Sterntopologie
(2) $l(n) = \sum ... = (n * (n-1))/2$ bei vollvermaschter Topologie

d) (1) LAN
- Reichweite: 10m
- Reaktionszeit: niedrig
- Datenrate: hoch
- Topologien: Sterntopologie

(2) MAN
- Reichweite: 10km
- Reaktionszeit: mittel
- Datenrate: mittel
- Topologien: hierarchische Topologie

(3) WAN
- Reichweite: 100km - 10.000km
- Reaktionszeit: hoch
- Datenrate: niedrig
- Topologien: Vollvermaschte Topologie

2.1.2

a) Dienst und Protokoll

- siehe Musterlösung

b) OSI Schichtenmodell

- Schichtenmodell siehe Folie 1.8ff
- Protokoll:
  - ist eine Sprache zur horizontalen Kommunikation zwischen Prozessen derselben Schicht auf verschiedenen Hosts
- Dienst
  - dient der vertikalen Kommunikation zwischen zwei Schichten auf einem Host
- Aufteilung des Bitstroms: Schicht 2 Sicherungsschicht
- Ende-zu-Ende Kommunkation: Schicht 4 Transportschicht
- Wegewahl: Schicht 3 Vermittlungsschicht

c) keine inhaltliche Bearbeitung, sondern nur Informationsweiterleitung

2.1.3

a)
- siehe Folie 1.15;
- Initiator (Prozess A), ...
- Responder (Prozess B), ...

b) (1) Zustände bestimmen
- idle
- connected
- prepare(Initiator)
- prepare(Responder)

(2) Übergänge bestimmen (Knoten, Pfad, Knoten)
- (idle, conReq, prep(Init))
- (idle, ConInd, prep(Resp))
- (prep(Resp), conRsp, connected)
- (prep(Init), conCnf, connected)
- (connected, dataRep/dataInd, connected)
- (prep(Resp)/prep(Init)/connected, disRep/disInd, idle)

c) (1) Ablaufdiagramm

- c1) + zeitlicher Ablauf
- c2) - es werden n Diagramme benötigt
- c3) -

(2) Zustandsdiagramm

- c1) -
- c2) + alle Abläufe in einem Diagramm darstellbar
- c3) +

2.1.4

a) siehe Folie 1.10

(1) $PDU(N) = SDU(N-1)$

(2) $IDU(N) = ICI(N) + SDU(N)$

b) Seitenaufruf: http://www.heise.de/software

(1) httpRequest

i. GET/software/http/1.1

ii. Host: www.heise.de

(2) ICI

i. ip: 193.99.144.85 port:80

(3) SDU

i. GET/software/http/1.1

ii. Host: www.heise.de

(4) IDU

i. ICI

ii. SDU

(5) TCP-PDU

i. src:80, dest:80,...

ii. SDU

iii. Data

c)

$$b_0 = 125\frac{\text{Mbit}}{\text{s}}$$
$$b_1 = b_0 \cdot 0,8$$
$$b_2 = b_1\frac{(55+99)0,01}{2}$$
$$b_3 = b_2\frac{(57+99)0,01}{2}$$
$$b_4 = b_3\frac{(23+99)0,01}{2} = 36,4\frac{\text{Mbit}}{\text{s}}$$
$$b_4 = b_{goodput}$$
$$b_{extra} = b_2\frac{(23+99)0,01}{2} = 46,7\frac{\text{Mbit}}{\text{s}}$$

2.2 Bitübertragungsschicht

2.2.1 Nyquist-Theorem

a)

- Bandbreite B [Hz]
- Signalrauschabstand SNR
- Abschneidefrequenzen in niedrigen Bereichen → Differenz = Bandbreite
- Signalstufen S [1], digitaler Signal: $S = 2$
- Signalrate SR [Hz]
- Bitrate b $[\frac{\text{Bit}}{\text{s}}]$

$b = SR \cdot ld(S)$

- Nyquist 1 (rauschfrei) $b < 2 \cdot B \cdot ld(2)$
- Nyquist 2 (verrauscht) $b < B \cdot ld(1 + SNR)$
- $SNR_{dB} = 10 \cdot log_{10}(SNR)$

Informationsgehalt $I = ld(16) = 4$ [Bit]f
Signalrate $SR = \frac{b}{ld(S)} = \frac{9600\frac{\text{Bit}}{\text{s}}}{4\ [\text{Bit}]}$

b)

$$SNR_{dB} = 12\text{dB}$$
$$SNR = 10^{\frac{SNR_{dB}}{10}} = 15$$
$$B > \frac{b}{(d(1+SNR))} = \frac{9600\frac{\text{Bit}}{\text{s}}}{ld(1+15)} = 2400\text{Hz}$$

c)

$$B > 963Hz \rightarrow B > 1200Hz\text{Nyquist 1}$$

2.2.2 Pulsecodemodulation

a)
- PCM: analoges Signal → Tiefpass (f_g) → Abtastung (zeitdiskret, (f_a)) → Quantisierung (wertdiskret)
- $f_a > 2f_g$

gegeben:

$$B = 60 \text{ kHz}$$
$$QS = 1024$$
$$S = 2$$

gesucht:

$$\begin{aligned} b &< 2 \cdot B \cdot ld(S) \\ &= 2 \cdot 60 \text{ kHz} \cdot 1 \text{ [Bit]} \\ &= \frac{120 \text{ kBit}}{\text{s}} \end{aligned}$$

$$\begin{aligned} f_a &= \frac{b}{(d(QS))} \\ &= \frac{b}{10 \text{ [Bit]}} \\ &= 12 \text{ kHz} \\ f_g &< \frac{f_a}{2} \\ &= 6 \text{ kHz} \end{aligned}$$

b)
- 8 Bit: leises Hintergrundrauschen
- 4 Bit: starkes Rauschen
- 1 Bit: extremes Rauschen

2.2.3 Modulation

a) Tabelle zeichnen siehe VL

(1) Amplitudenmodulation
(2) Frequenzmodulation
(3) Phasenmodulation

b)
- Realisierungsaufwand
  (1) sehr gut

(2) gut
(3) schlecht

- Störsicherheit
  (1) schlecht
  (2) gut
  (3) sehr gut

2.2.4 Leitungskodierung

a) Leitungskodierung:
Anpassung des Datenstroms an das eingesetzte Übertragungsmedium
Ziel:

- Erkennung von Leitungsunterbrechungen
- Synchronisation ($\rightarrow$ Takt)

b) einfacher Manchesterkodierung

- Leitungskode (Takt kann herausgelesen werden, Leitungen können unterbrochen werden)
- jedes Bit in zwei Intervalle teilen
- $0 \rightarrow$ low, high
- $1 \rightarrow$ high, low

c) siehe b)
Übertragungsrate = $\frac{1}{2}$ Signalrate

2.2.5 Multiplex

a) siehe Musterlösung

- synchrones Zeitmultiplex
  - feste Zeitschlitze - Taktung nötig
  - fester Zeitabschnitt für jeden Sender
  - Zuordnung zu virtuellem Kanal durch Position im Übertragungskanal
- asynchrones Zeitmultiplex
  - feste Zeitschlitze - Taktung nötig
  - Senden nach Bedarf
  - Channel Identifier für Zuordnung zu virtuellem Kanal

b) gegeben:

$$B_{\text{Gesamt}} = 49 \text{ MHz}$$
$$B_{\text{Abstand}} = 1,5 \text{ MHz}$$
$$B_{\text{Kanal}} = 5,5 \text{ MHz}$$

gesucht: Anzahl n der möglichen Fernsehkanäle:

$$\begin{aligned} 49 \text{ MHz} &= 5,5 \text{ MHz} \cdot n + 1,5 \text{ MHz} \cdot (n-1) \\ 49 \text{ MHz} &= (5,5 + 1,5) \text{ MHz} \cdot n - 1,5 \text{ MHz} \\ 50,5 &= 7n \\ n &= \frac{50,5}{7} = 7,21 \end{aligned}$$

$\rightarrow$ Es können 7 Kanäle angeboten werden.

2.3 Netztechnologien 1

2.3.1 Ethernet

a) CSMA/CD

(1) t_0: A beginnt ein Frame zu senden

(2) $t_0 + \tau - \epsilon$ B beginnt ein Frame zu senden

(3) $t_0 + \tau$ B erkennt Kollision

(4) $t_0 + \tau - \epsilon + \tau$ A erkennt die Kollision

$\Rightarrow t_s > 2\tau$ mit t_s Sendezeit für ein Frame

$$\begin{aligned}
\tau &= \frac{d_{ges}}{v_{phy}} \\
&= \frac{|\text{Repeater}| \ + 1 \cdot d}{v_{phy}} \\
&= \frac{5 \cdot 500 \text{ m}}{\frac{200000\text{km}}{s}} \\
&= 12,5\mu s \\
&\rightsquigarrow 2\tau = 25\mu s \\
\text{Sendezeit} &= \frac{\text{Rahmenlänge}}{Bitrate} \\
t_s &= \frac{F}{b} \\
F &= t_s \cdot b > 2\tau \cdot b \\
&= 25\mu s \cdot 10\frac{\text{MBit}}{s} \\
&= 250\text{Bit}
\end{aligned}$$

b) Ethernet Flow Control Switches kümmern sih um die Behandlung von potentiellen Kollisionen EFC dient ... siehe Lösungen

2.3.2 Switches

|Präambel (8)|Ziel (6)|Quelle (6)|Typ/Size (2)|
Daten (1500)|CRC (4)|Interframe Gap (12)| $\rightsquigarrow$ () in Byte

a) Store-and-Foreward Switches
t_s und t_V werden vernachlässigt

$$\begin{aligned}
& t_s + t_F + t_V + t_s + t_F \\
t_F = \frac{SL + F}{b} &= \frac{(8 + 1518) \cdot 8\text{Bit}}{100 \cdot 10^6 \frac{\text{Bit}}{s}} \\
&= 122,08\mu s \\
t_{ges} &= 4t_F = 488,32\mu s
\end{aligned}$$

b) Virtual-Cut-Through Switches

$$\begin{aligned} & t_s + t_H + t_s + t_F \\ t_{ges} &= 3t_H + t_F \\ &= 122,08\mu s + 3 \cdot \frac{14 \cdot 8 \text{ Bit}}{100 \cdot 10^6 \frac{\text{Bit}}{s}} \\ &= 122,08\mu s + 3 \cdot 1,12\mu s \\ &= 125,44\mu s \end{aligned}$$

2.3.3 Transparent Bridges

a) Wegewahltabellen siehe Lösungen

b) Bridge Informationen:
X|Y :
X = kommt von welchem Rechner,
Y = kommt an welchem Port an

c) Probleme

- bei Zyklen (Pakete werden im Kreis weitergereicht

d) Spanning Tree Protocol siehe Lösungen

2.3.4 802.11 WLAN

a) Wkt., dass ein Bit falsch ist:

$$e_{\text{Bit}} = 3 \cdot 10^{-6}$$

Wkt., dass ein Bit richtig ist:

$$p_{\text{Bit}} = 1 - e_{\text{Bit}}$$

Rahmen mit 1500 Byte Größe:

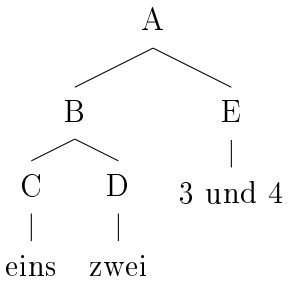
$$\begin{aligned} p_{Frame} &= p_{Bit} \cdot \ldots \cdot p_{Bit} = (p_Bit)^{1500 \cdot 8} \\ &= (1 - 3 \cdot 10^{-6})^{12000} = 0,96464 \end{aligned}$$

$$e_{\text{Frame}} = 1 - p_{Frame} = 0,035$$

$\rightarrow$ im Mittel 3,5 prozent der Frames fehlerhaft.

b)

$$\begin{aligned} p_{Frame} &= (p_{Bit})^{(64 \cdot 8)} = 0,99847 \\ \rightarrow e_{Frame} &= 0,00153 \end{aligned}$$


A
B
E
C
D
3 und 4
eins
zwei

Teil II

Theoretical Informatic and Logic

Kapitel 3

Vorlesung

3.1 Prädikatenlogik erster Stufe

- Syntax
  - Ein Alphabet der Prädikatenlogik besteht aus ... (2)
  - forall heist universeller Quantor, exists heißt existenzieller Quantor
  - Funktions- und Relationssymbolen ist eine Stelligkeit n el N
  - Nullstellige Funktionssymbole werden als ... (3)
- Terme
  - Definition 4.2 prädikatenlogische Terme (4)
  - Ein Term ist abgeschlossen oder grundinstanziiert, wenn in ihm keine Variablen vorkommen
  - Die Menge der abgeschlossenen Terme wird mit T (F) bezeichnet
- Prädikatenlogische Atome (5)
- Prädikatenlogische Formeln (6)
  - prädikatenlogische Formeln
- Strukturelle Rekursion
  - Rekursionssätze lassen sich für $T(F,\ V)$ und $L(R,F,V)$ formulieren
  - Es gibt genau eine Funktion *foo* die die folgenden Bedingungen erfüllt: (7)
    - Rekursionsanfang
    - Rekursionsschritt
  - Beispiele (8/9)

3.2 Prädikatenlogik erster Stufe

- Strukturelle Induktion
  - Induktionssätze lassen sich für T(F,V) und L(R,F,V) formulieren
  - jeder Term besitzt die Eigenschaft E, wenn: (10)
  - analog für prädikatenlogische Formeln
- Aufgabe (11)
  - Beweisen Sie, dass $\forall F \in L(R, F, V)$ die Aussage $l'(m(F)) \geq l(F)$ gilt
- Teilterme und Teilformeln (12)
  - Die Def. 3.8 lässt sich auf Terme und Formeln übertragen
  - Beispiel
- Freie und gebundene Vorkommen einer Variablen (13)
  - Def. 4.5 Die **freien Vorkommen einer Variablen** in einer prädikatenlogischen Formel sind wie folgt definiert: (13)
- Abgeschlossene Terme und Formeln (14)
  - nach Def. 4.2: Ein abgeschlossener Term ist ein Term, in dem keine Variable vorkommt
  - Def. 4.6 Eine abgeschlossene Formel (oder kurz ein Satz) der Sprache *L(R,F,V)* ist eine Formel der Sprache *L(R,F,V)*, in der jedes Vorkommen einer Variablen gebunden ist
- Substitutionen (19)
  - Def. 4.7: Eine **Substitution** ist eine Abbildung $\sigma : V \to T(F, V)$, die bis auf endlich viele Stellen mit der Identitätsabbildung übereinstimmt
  - Beispiel
- Instanzen
  - Statt $\sigma(X)$ schreiben wirn in der Folge $X\sigma$
  - Def. 4.8: Sei sigma eine Substitution $\sigma : V \to T(F, V)$ kann wie folgt zu einer Abbildung $\sigma dach : T(F, V) \to T(F, V)$ erweitert werden: (25)
  - Grundinstanz
  - Proposition
- Komposition von Substitutionen

- Def. 4.10: Seien σ und θ zwei Substitutionen Die Komposition $\sigma\theta$ von σ und θ ist die Substitution: (30)
- Aufgaben

3.3 Syntax/Substitutionen

3.3.1 Komposition von Substitutionen

Korollar 4.11

Für jede Substitution σ gilt $\epsilon\sigma = \sigma = \sigma\epsilon$

Proposition 4.12

Seien σ und θ Substitutionen. Für jeden term t gilt $t(\hat{\sigma\theta}) = (t\hat{\sigma})\theta$
Beweis Strukturelle Induktion über t $\rightarrow$ Übung

Proposition 4.13

Sei $t \in T(F, V)$ und seien σ, θ sowie λ Substitutionen. Dann gilt:

- $t((\sigma\hat{\theta})\lambda)$
- $\sigma\theta)\lambda = \sigma(\theta\lambda)$

Beweis siehe Folien (19)

3.3.2 Beschränkung von Substitutionen

Definition 4.14

Sei σ eine Substitution. Dann ist

$$\sigma_x = \begin{cases} \sigma & \text{wenn } X \notin \text{dom}(\sigma) \\ \sigma/\{X \mapsto t\} & \text{wenn } X \mapsto t \in \sigma \end{cases}$$

Proposition 4.15

Sei σ eine Substitution und t ein Term, in dem die Variable X nicht vorkommt.
Dann gilt: $t\sigma = t\sigma_X$

3.3.3 Anwendung von Substitutionen auf Formeln

Definition 4.16

Die Anwendung einer Substitution σ auf eine Formel ist induktiv über den Aufbau prädikatenlogische Formel wie folgt definiert:

- $p(t_1, \ldots, t_n)\sigma = p(t_1\sigma, \ldots, t_n\sigma)$
- $(\neg F)\sigma = \neg(F\sigma)$
- $(F \circ G)\sigma = (F\sigma \circ G\sigma)$ für jeden binären Junktor $\circ$
- $((QX)F)\sigma = (QX)(F\sigma_X)$ für jeden Quantor Q

Beobachtung

Bei der Anwendung einer Substitution auf eine Formel werden nur frei vorkommende Variablen ersetzt
Beweis: Übung

3.3.4 Substitutionen und Formeln

Definition 4.17

Eine Substitution σ ist genau dann frei für eine prädikatenlogische Formel F, wenn sie sich gemäß der folgenden bedingungen als frei erweist:

- σ ist frei für F, wenn F ein Atom ist
- σ ist frei für $\neg F$ **gdw** σ ist frei für F
- σ ist frei für $(F \circ G)$ **gdw** σ ist frei für F und σ ist frei für G
- σ ist frei für $(QY)F$ **gdw** σ_Y ist frei für F und für jede von Y verschiedene und in F frei vorkommende Variable X gilt: Y kommt in $X\sigma$ nicht vor

3.3.5 Satz 4.18

Satz 4.18

Wenn die Substitution σ frei für die prädikatenlogische Formel F und die Substitution θ frei für $F\sigma$ ist, dann gilt: $F(\sigma\theta) = (F\sigma)\theta$

Beweis Satz 4.18

Strukturelle Induktion über F

- **IA** F ist Atom der Form $p(t_1, \ldots, t_n)$

$$\begin{aligned}
&p(t_1, \ldots, t_n)(\sigma\theta) \\
&= p(t_1(\sigma\theta), \ldots, t_n(\sigma\theta)) && \text{Def 4.16} \\
&= p((t_1\sigma)\theta, \ldots, (t_n\sigma)\theta) && \text{Prop 4.12} \\
&= p(t_1\sigma, \ldots, t_n\sigma)\theta && \text{Def 4.16} \\
&= p(t_1, \ldots, t_n)\sigma)\theta && \text{Def 4.16}
\end{aligned}$$

- **IH** Das Resultat gilt für F
- **IS**
  - **Fall** $\neg F$

    Sei σ frei für $\neg F$ und θ frei für $(\neg F)\sigma$
    Da σ frei für $\neg F$ ist, ist σ auch frei für F
    Da θ frei für$(\neg F)\sigma$ und $(\neg F)\sigma = \neg(F\sigma)$ ist, ist θ auch frei für $F\sigma$
    $((\neg F)\sigma)\theta = (\neg(F\sigma))\theta = \neg((F\sigma)\theta) =_{(IH)} \neg(F(\sigma\theta)) = (\neg F)\sigma\theta$

  - **Fall** $(F \circ G) \rightsquigarrow$ Übung
  - **Fall** $(\forall X)F$

    Sei σ frei für $(\forall X)F$ und θ frei für $((\forall X)F)\sigma$
    Da σ frei für $(\forall X)F$ ist, ist σ_X frei für F
    Da θ frei für $((\forall X)F)\sigma = (\forall X)(F\sigma_X)$ ist, ist θ_X frei für $F\sigma_X$
    Hilfsaussage $F(\sigma_X\theta_X) = F(\sigma\theta)_X$
    Dann gilt:

$$\begin{aligned}
&(((\forall X)F)\sigma)\theta \\
&= ((\forall X)(F\sigma_X))\theta && \text{Def 4.16} \\
&= (\forall X)((F\sigma_X)\theta_X) && \text{Def 4.16} \\
&= (\forall X)(F(\sigma_X\theta_X)) && \text{IH} \\
&= (\forall X)(F(\sigma\theta)_X) && \text{Hilfsaussage} \\
&= ((\forall X)F)(\sigma\theta) && \text{Def 4.16}
\end{aligned}$$

  - **Fall** $\exists X)F \rightsquigarrow$ Übung

3.3.6 Beweis Hilfsaussage aus Satz 4.18

Unter den genannten Bedingungen gilt $F(\sigma_X\theta_X) = F(\sigma\theta)_X$
Beweis Da in F nur frei vorkommende Variablen ersetzt werden, genügt es zu zeigen, dass für jede frei in F vorkommende Variable Y gilt: $Y(\sigma_X\theta_X) = Y(\sigma\theta)_X$

- **Fall** $Y = X$

$$Y(\sigma_X\theta_X) = Y = Y(\sigma\theta)_X$$

- **Fall** $Y \neq X$

  $Y\sigma = Y\sigma_X$ und $Y(\sigma\theta) = Y(\sigma\theta)_X$
  Da σ frei für $(\forall X)F$ ist, kommt die Variable X in $Y\sigma$ nicht vor
  Deshalb ist $(Y\sigma)\theta = (Y\sigma)\theta_X$
  Dann gilt:

$Y(\sigma_X\theta_X)$	
$= (Y\sigma_X)\theta_X$	Prop 4.12
$= (Y\sigma)\theta_X$	$(X \neq Y)$
$= (Y\sigma)\theta$	X kommt in Y nicht vor
$= Y(\sigma\theta)$	Prop 4.12
$= Y(\sigma\theta)_X$	$(X \neq Y)$

3.3.7 Varianten

Definition 4.19

Seien E_1 und E_2 zwei Terme oder zwei prädikatenlogische Formeln. E_1 und E_2 heißen Varianten, wenn es Substitutionen σ und θ gibt, so dass $E_1 = E_2\sigma$ und $E_2 = E_1\theta$. In diesem Fall wollen wir E_1 auch als Variante von E_2 und E_2 als Variante von E_1 bezeichnen.
Wenn E_1 und E_2 Varianten sind und die in E_2 vorkommenden Variablen im bisherigen Kontext nicht verwendet wurden, dann ist E_2 eine neue Variante von E_1.

3.4 Semantik

3.4.1 Relationen und Funktionen

- Sei D eine Menge ...
- Relationen ...
- **Notation**: Anstelle von (d) schreibt man häufig kurz d
- Funtionen $(+, \circ,)$

3.4.2 Interpretationen

Definition 4.20

Eine prädikatenlogische Interpretation I für eine prädikatenlogische Sprache $L(R, F, V)$ besteht aus einer nichtleeren Menge D und einer Abbildung $\cdot^I$, die die folgenden Bedingungen erfüllt:

- Jedem n-stelligen Funktionssymbol $g \in F$ wird eine n-stellige Funktion $g^I : D^n \to D$ zugeordnet
- Jedem n-stelligen Prädikatssymbol $p \in R$ wird eine n-stellige Relation $p^I \subseteq D^n$ zugeordnet

D wird Grundbereich oder auch Domäne der Interpretation genannt

Definition 4.21

Eine Variablenzuordnung bezüglich einer Interpretation $I = (D, \cdot^I)$ ist eine Abbildung $Z : V \to D$ Das Bild einer Variablen X unter Z bezeichnen wir mit X^Z Sei Z eine Variablenzuordnung und $d \in D$, mit $\{X \to d\}Z$ bezeichnen wir die Variablenzuordnung für die gilt:

$$Y^{\{X \mapsto d\}Z} = \begin{cases} d & \text{wenn } Y = X \\ Y^Z & \text{sonst} \end{cases}$$

Definition 4.22

Sei $I = (D, \cdot^I)$ eine Interpretation und Z eine Variablenzuordnung bezüglich I. Die Bedeutung $t^{I,Z}$ eines Terms $t \in T(F, V)$ ist wie folgt definiert:

- Für jede Variable $X \in V : X^{I,Z} = X^Z$
- Für jeden Term der Form $g(t_1, \ldots, t_n)$ ist

  $$[g(t_1, \ldots, t_n)]^{I,Z} = g^I(t_1^{I,Z}, \ldots, t_n^{I,Z})$$

  wobei $g/n \in F$ ist und $t_1, \ldots, t_n \in T(F, V)$ sind

Definition 4.23

Sei $I = (D, \cdot^I)$ eine Interpretation und Z eine Variablenzuordnung bezüglich I und I und Z weisen jeder Formel $F \in L(R, F, V)$ einen Wahrheitswert $F^{I,Z}$ wie folgt zu:

$$[p(t_1, \ldots, t_n)]^{I,Z} = \top \text{ gdw } (t_1^{I,Z}, \ldots, t_n^{I,Z} \in p^I$$
$$[\neg F)]^{I,Z} = \neg^*(F^{I,Z})$$
$$[(F \circ G)]^{I,Z} = (F^{I,Z} \circ^* G^{I,Z}) \text{ für alle binären Juntkoren } \circ$$
$$[(\forall X)F]^{I,Z} = \top \text{ gdw } F^{I,\{X \mapsto d\}Z} = \top \text{ für alle } d \in D$$
$$[(\exists X)F]^{I,Z} = \top \text{ gdw } F^{I,\{X \mapsto d\}Z} = \top \text{ für alle } d \in D$$

Proposition 4.24

Wenn $F \in L(R, F, V)$ abgeschlossen ist, dann gilt $F^{I,Z} = F^{I,Z^I}$ für jede Interpretation I und alle Variablenzuordnungen Z und Z^I bezüglich I

Lemma 4.25

Seien s, t Terme, G eine Formel, Y eine Variable, $I = (D, \cdot^I)$ eine Interpretation, Z eine Variablenzuordnung bzgl. I und $d \in D$. Wenn $[t]^{I,Z} = d$ ist, dann gilt:

$$[s\{Y \mapsto t\}]^{I,Z} = [s]^{I,\{Y \mapsto d\}Z}$$
$$[G\{Y \mapsto t\}]^{I,Z} = [G]^{I,\{Y \mapsto d\}Z}, \text{ wenn } \{Y \mapsto t\} \text{ frei für } g \text{ ist}$$

Beweis Induktion über den Aufbau von s bzw. $G \rightsquigarrow$ Übung

3.4.3 Herbrand-Interpretationen

Im Folgenden nehmen wir an, dass F mindestens ein Konstantensymbol enthält. Ist das nicht der Fall, dann fügern wir zu F ein Symbol $a/0$ hinzu

Definition 4.26

Sei F eine Menge von Funktionssymbolen, in der mindstens ein Konstantensymbol vorkommt. Eine Interpretation $I = (D, \cdot^I)$ für eine prädikatenlogische Sprache $L(R, F, V)$ ist eine Herbrand-Interpretation, wenn die folgenden Bedingungen erfüllt sind:

$$D = T(F) \text{ (D wird Herbrand-Universum genannt)}$$
$$\text{Für jeden abgeschlossenen Term } t \in T(F) \text{ gilt } t^I = t$$

3.5 Modelle

Herbrand-Interpretationen und Formeln (41)

Aufgabe (42)

3.5.1 Modelle für abgeschlossene Formeln

Definition 4.27

Sei $I = (D, \cdot^I)$ eine Interpretation und $F \in L(R, F, V)$ ein Satz I ist ein Modell für F, symbolisch $I \vDash F$, wenn gilt: $F^I = \top$
Viele aus der Aussagenlogik bekannte Begriffe und REsultate lassen sich auf die Prädikatenlogik übertragen. Zum Beispiel:

- Allgemeingültigkeit, Erfüllbarkeit, Widerlegbarkeit und Unerfüllbarkeit
- z.B.: Ein Satz F ist allgemeingültig gdw alle Interpretationen Modelle für F sind
- Satz 3.14 erweitert: Ein Satz F ist allgemeingültig gdw $\neg F$ ist unerfüllbar
- Satz 3.17 erweitert: Seien $F, F_1, \ldots, F_n$ Sätze
  $\{F_1, \ldots, F_n\} \vDash F$ gdw $\vDash (\langle F_1, \ldots, F_n \rangle \to F)$

Definition 4.28

Ein Satz F ist eine (prädikatenlogische) Konsequenz der Menge G von Sätzen, symbolisch $G \vDash F$, gdw. jedes Modell für alle Elemente aus G auch Modell für F ist.

Aufgaben (45)

Aussagenlogik - Prädikatenlogik

- Wenn alle Relationssymbole in R nullstellig sind, dann ist die Prädikatenlogik äquivalent zur Aussagenlogik
- Wenn in der Formeln keine Variablen vorkommen, dann ist die Prädikatenlogik äquivalent zur Aussagenlogik

3.5.2 Modelle für nicht-abgeschlossene Formeln

Definition 4.29

Sei $G \in L(R, F, V)$ und $fv(G) = \{X_1, \ldots, X_n\}$

$$ucl(G) = (\forall(X_1, \ldots, X_n)G \text{ ist der universelle Abschluss von } G$$
$$ecl(G) = (\exists X_1, \ldots, X_n)G \text{ ist der existenzielle Abschluss von } G$$

Definition 4.30

$$I \models_u \text{ wenn } I \models ucl(G)$$
$$I \models_e \text{ wenn } I \models ecl(G)$$

Proposition 4.31

Für alle Sätze G gilt:

$$ucl(G) = G = ecl(G)$$
$$I \models G \text{ gdw } I \models_u G \text{ gdw } I \models_e G$$

Universeller Abschluss: Einige Eigenschaften (49)

$$\models_u ((\forall X)p(X) \rightarrow p(X))$$
$$\not\models_u (p(X) \rightarrow (\forall X)p(X)$$

3.6 Äquivalenz und Normalform

3.6.1 Semantische Äquivalenz

Definition

Zwei prädikatenlogische Formeln F und G heißen **semantisch äquivalent**, symbolisch $F \equiv G$, wenn $F^{I,Z} = G^{I,Z}$ für alle Interpretationen I und alle Variablenzuweisungen Z bezüglich I

Satz 3.19

Für prädikatenlogische Sätze, Formeln gilt:

$$F \equiv G \leftrightarrow F^I = G^I$$

Satz 4.32

Seien F und G prädikatenlogische Formeln. Dann gilt:

$$\neg(\forall X)F \equiv (\exists X)\neg F$$
$$\neg(\exists X)F \equiv (\forall X)\neg F$$
$$((\forall X)F \wedge (\forall X)G) \equiv (\forall X)(F \wedge G)$$

$$fill$$

Beweis 1

siehe Script, die anderen in Eigenarbeit

Definition 4.33

Die in einer Prädikatenlogischen Formel F vorkommenden Variablen sind **auseinander dividiert**, wenn keine zwei in F vorkommenden Quantoren die gleiche Variable binden und keine Variable sowohl frei als auch gebunden vorkommt

Definition 4.33

Zu jeder prädikatenlogischen Formel gibt es eine semantisch äquivalente Formel, in der die Variablen auseinander dividiert sind

Vereinbarung

In der Folge nehmen wir an, dass die Variablen auseinander dividiert sind.

3.6.2 Pränexnormalform

Definition 4.35

Eine Formel G ist in **Pränexnormalform**, wenn sie von der Form $(Q_1X_1)\dots(Q_nX_n)F$ ist, wobei $Q_i \in \{\forall, \exists\}$, $\leq i \leq n$ und $n \geq 0$ ist, $X_1, \dots, X_n$ Variablen sind und in F selbst kein Quantor mehr vorkommt. Wir nennen F auch Matrix von G

Proposition 4.36

Es gibt einen Algorithmus, der einen Satz F in der Prädikatenlogik in einen semantisch äquivalenten Satz F' in Pränexnormalform transformiert

Transformation in Pränexnormalform

Solange die Formel F nciht in Pränexnormalform ist, wende eine der folgenden Regeln an:

$$\frac{\neg(\forall X)F}{(\exists X)\neg F} \qquad \frac{\neg(\exists X)F}{(\forall X)\neg F}$$

$$\frac{((QX)F \wedge G)}{(QX)(F \wedge G)} \qquad \frac{(F \wedge (QX)G)}{(QX)(F \wedge G)}$$

$$\frac{((QX)F \vee G)}{(QX)(F \vee G)} \qquad \frac{(F \vee (QX)G)}{(QX)(F \vee G)}$$

3.6.3 Skolem-Normalform

Idee

Wir beseitigen alle existenziellen Quantoren

Definition 4.37

Sei $L = L(R, F, V)$ eine prädikatenlogische Sprache. Sei F_S eine abzählbar unendliche Menge von Funktionssymbolen, so dass $F_S \cap F = \emptyset$ und F_S für jede Stelligkeit abzählbar unendlich viele Funktionszeichen enthält. Die Elemente aus F_S werden **Skolem-Funktionssymbole** genannt. Wir betrachten nun die Sprache $L(R, F \cup F_S, V)$

Notation

0-stellige Skolem-Funktionssymbole werden häufig auch **Skolem-Konstantensymbole** genannt

Definition 4.38

Eine prädikatenlogische Formel ist in **Skolem-Normalform**, wenn sie von der Form $\forall X_) \dots (\forall X_n) f$ ist, wobei $n \geq 0$ ist, $X_1, \dots, X_n$ Variablen sind und in F selbst kein Quantor mehr vorkommt

Transformation in Skolem-Normalform

Sei F Formel in Pränexnormalform, deren variablen auseinander dividert sind. Solange F nicht in Skolem-Normalform ist, wende die folgende Regel an

$$\frac{(\forall X_1) \dots (\forall X_n)(\exists Y) G}{(\forall X_1) \dots (\forall X_n) G\{Y \mapsto f(X_1, \dots, X_n)\}}$$

Satz 4.39

Sei F' eine Skolem-Normalform des Satzes F. F ist erfüllbar gdw F' ist erfüllbar.
$\rightarrow$ Die Tranformation in Skolem-Normalform ist erfüllbarkeitserhaltend

Beweis Satz 4.39

- Anname: F in Pränexnormalform; Variablen sind auseinander dividiert
- **Hilfsaussage**: Sei F ein Satz in Pränexnormalform, in der die Variablen auseinander dividert sind. Sei F' durch einmalige Anwendung der Ersetzungsregel auf F entstanden. Dann gilt: F ist erfüllbar gdw. F' ist erfüllbar
- Beweis Hilfsaussage $\rightsquigarrow$ Übung

- Sei E die folgende Zusicherung: F' ist ein Satz in Pränesnormalform, in der alle Variablen auseinander dividiert sind und F' ist erfüllbar gdw F ist erfüllbar
- mit $F = F'$ ist E vor Eintritt in die Schleife erfüllt
- Gemäß der Hilfsaussage ist E eine Schleifeninvariante
- Nach **Satz 3.30** gilt E dann auch nach Verlassen der Schleife
- Die Schleife wird nur verlassen wenn F' in Skolem-Normalform ist

3.6.4 Klauselform

Kapitel 4

Übung

tobias.philipp@tu-dresden.de
Fr. 14:00 2005/2006

4.1 Prädikatenlogik - Syntax

4.1.1 Konstruktion von Teiltermen

a) Bestimme für den Term $t = h(f(Y, X), Y, g(a))$ gemäß obiger Definition 2 die Mengen $K_n(t)$ für alle $n \in \mathbb{N}$ und geben Sie für alle Terme $s \in K_n(t)$ die zugehörige Konstruktion an.

$K_0(t) = \{t\}$	$[t]$
$K_1(t) = \{f(Y, X), X, g(a)\}$	$[t, f(Y, X)], [t, X], [t, g(a)]$
$K_2(t) = \{Y, X, a\}$	$[t, f(Y, X), Y], [t, f(Y, X), X], [t, g(a), a]$
$K_3(t) = \emptyset$	
$\Rightarrow K(t) = \{t, f(Y, X), X, g(a), Y, a\}$	

b) Zeigen Sie, dass $T_t = K(t)$ für beliebige $t \in T(F, V)$ gilt.

z.Z.: $T_t = \{t, f(Y, X), X, g(a), Y, a\}$
1. Idee: $T_t = K(t) \forall t \in T(F, V)$
2. Idee: Strukt. Induktion

c) Strukturelle Induktion

- IA

  Fall a) t ist von der Form $X \in V$
  Wir wissen, dass $K_0(t) = X$ Außerdem gilt $K_n(t) = \emptyset \forall n \geq 1$.
  Aus der Def. von $K(t)$, wissen wir dann dass $K(t) = \{X\}$
  z.Z.: $T_t = \{X\}$
  $\Rightarrow \{X\}$ erfüllt Bed 1 und 2 der Def 1 und Minimalität
  1. ist offensichtlich erfüllt
  2. ist erfüllt, weil die Vorbedingung immer falsch ist
  Minimalität über $\emptyset$
  $\Rightarrow K(t) = T_t$
  Fall b) t ist v.d.F. Atom $\in F$ **analog**

- IV Die Aussage gelte für $t_1, \ldots, t_n)$
- IS z.Z Die Aussage gilt für $f(t_1, \ldots, t_n)$

$$
\begin{aligned}
T_{f(t_1,\ldots,t_m)} &= K(f(t_1, \ldots, t_m)) \\
&= \bigcup_{n=0}^{\infty} K_n(f(t_1, \ldots, t_m))
\end{aligned}
$$

zuerst $\supseteq$

Sei $s \in \bigcup_{n=0}^{\infty} K_n(f(t_1, \ldots, t_m))$ Dann ex. ein l sodass
$s \in K_l(f(t_1, \ldots, t_m)[s_0, \ldots, s_l]$
Dann gibt es Konst der Länge $l-1$ von s aus $t_i, i \in \{1, \ldots, m\}$
$s \in K_{l-1}(t_i) \forall i$
$\Rightarrow s \in K(t_i)$
$\Rightarrow s \in T_{t_i}$nach I.V.
Es fehlt $\subseteq$

4.1.2 Über Nachbarn

a) Definieren Sie ...

$$(\forall X)\Big((\exists Y)(m(X) \wedge e(X,Y)) \leftrightarrow \text{ vater}(X)\Big)$$
$$(\forall X)(\exists Y)(m(X) \wedge e(X,Y)) \leftrightarrow \text{ vater}(X)$$

b) Drücken Sie ...

$$(\forall X)\neg n(X,X)$$
$$(\forall X)\neg(d(X,X)\wedge n(X,X))$$

4.2 Substitutionen

4.2.1 Substitutionskomposition ist eine Substitution

Siehe Lösungen Übungsbuch

z.Z.: $\sigma\theta$ ist eine Funktion
1. Fall: Wenn $X \notin dom(\sigma\theta)$
$\nexists$ Paar $(X \mapsto t\theta) \in M_1$ mit $X \neq t\theta$
$\nexists$ Paar $(X \mapsto s) \notin M_2$
Dann gibt es ein solches Paar $(X \mapsto s) \notin (M_1 \cup M_2)$
X bildet auf sich selbst ab
2. Fall:
$\nexists(Paar)(X \mapsto t\sigma) \in M_1, X \neq t\theta$, und
$\exists$ Paar $(X \mapsto s) \in M_2$
z.Z.: $dom(\sigma\theta)$ ist endlich
$\{X|X \in U$ und $\sigma\theta(X) \neq X\}$
DEF.: $\sigma\theta \sim \{X \mapsto t\theta | X \mapsto t \in \sigma, X \neq t\theta\}$
$\cup \{Y \mapsto s | Y \mapsto s \in \theta, Y \notin dom(\sigma)\}$
$dom(\sigma\theta) \subseteq dom(\sigma) \cup dom(\theta)$
Die Vereinigung von endlichen Mengen ist endlich.
Da $dom(\sigma\theta)$ eine Teilmenge ist, folgt Endlichkeit.
Sei $X \in dom(\sigma\theta)$
Dann $\sigma\theta(X) \neq X$
Dann muss es ein Paar:
1.$(X \mapsto t) \in M_1$ oder
2.$(X \mapsto t) \in M_2$ aufgen können
1.$\exists X \mapsto s \in \sigma, t = s\theta$
$X \in dom(\sigma), X \in dom(\sigma) \cup dom(\theta)$
2.$X \mapsto t \in \theta, X \in dom(\theta), X \in dom(\sigma) \cup dom(\theta)$

4.2.2 Eingenschaften von Substitutionen

Proposition 4.12

a) Zeigen Sie, dass für alle Variablen $X \in V$ gilt:
$X(\widehat{\sigma\theta}) = (X\hat{\sigma})\hat{\theta}$

1. $\exists X \mapsto t \in \sigma\theta$

...

Fallunterscheidung siehe Lehrbuch

b) Strukturelle Induktion siehe Lehrbuch

- IA

Teil III

Computer Architecture

Kapitel 5

Vorlesung

5.1 Einführung

5.1.1 Big Data

"Big Data hat die Chance die geistige Mittelschicht in Hartz IV zu bringen"

5.2 Vorlesung

5.2.1 ZIH

- HAEC
- CRESTA Perfomrance optimization
- MPI correctness checking: MUST
- Architecture of the new system (HRSK-II)

5.2.2 Begriffe und Definitionen

- Der Begriff Rechnerarchitektur wurde von dem englischsprachigen Begriff computer architecture abgeleitet
- Computer architecture ist eine Teildisziplin des Wissenschaftsgebietes computer enginering, welches die überwiegend ingeniermäßige Herangehensweise beim Entwurf und der Optimierung von Rechnersystemen deutlich zum Ausdruck bringt.
- Zwei Deutungen des englischen Begriffs Ärchitecture"
- Zur Definition der Rechnerarchitektur

  - Architektur: Ausdruck insbesondere der Möglichkeiten der Programmierschnittstelle
    * Maschinenbefehlssatz
    * Registerstruktur
    * Adressierungsmodi
    * Unterbrechungsbehandlung
    * Ein- und Ausgabe-Funktionalität
  - Komponenten / Begriffsbildung
    * Hardwarestruktur
    * Informationsstruktur (Maschinendatentypen)
    * Steuerungsstruktur
    * Operationsprinzip

- Taxonomie
- Dreiphasenmodell zum Entwurf eins Rechnersystems
  - Bottom-up (Realisierung → Implementierung → Rechnerarchitektur)
  - Top-down (Rechnerarchitektur → Implementierung → Realisierung)
  - Rückwirkungen durch den technologischen Stand

5.3 VL

5.3.1 Modifiziertes Dreiphasenmodell zum Entwurf eines RS

Befelhlssatzarchitektur

Implementierung

Realisierung

5.3.2 Architektur-Definition (Tanenbaum)

Den Satz von Datentypen, Operationen und Merkmalen jeder Ebene bezeichnet man als ihre Architektur. Die Architektur betrifft die Aspekte, die für den Benutzer der jeweiligen Ebene sichtbar sind.

5.3.3 Architektur-Definition (Hennessy/Patterson)

We use the term instruction set architecture (ISA) to refer to the actual programmervisible instruction set in this book. the ISA serves as the boundary betweeen the software and hardware.
The implementation of a computer has two components: organization and hardware.

5.3.4 Einflusskomplexe

- Die Rechnerarchitektur steht in Wechselwirkung mit zahlreichen benachbarten Disziplinen
  - Betriebssysteme
    - Konzepte aus dem Bereich Betriebssysteme werden durch Rechnerkomponenten unterstützt (z.B.: Virtueller Speicher/ I/O-Instruktionen)
    - Andererseits werden durch moderne Rechnerarchitektur-Konzepte neue Anforderungen and die Betriebssysteme gestellt (z.B.: Erweiterung für Parallelarbeit)
  - Topologie
    - Ursprünglich Teilgebiet der Mathematik zur Untersuchung der Struktur von Punktmengen und Räumen einschließlich ihrer Klassifizierung
    - Daraus folgte: Gestaltung von Verbindungseinrichtungen in Multiprozessorsystemen (3-D Hypercube, D-Gitter)
  - Hardware-Entwurf
    - Die Sammlung von Anforderungen bildet die strukturelle und organisatorische Entwurfsspezifikation der Teilkomponenten eines Rechners
    - Darstellung: Very-High-Scale-Hardware-Description-Language (VHDL)
  - Compilerbau und Softwaretechnik
    - Codegenerator eines Compilers hänt von Architektur und Befehlssatz des Zielprozessors ab.
    - Intel-Titanium-Prozessor: EPIC-Architektur
  - Software-Entwurf
    - Nutzung unterschiedlicher Programmierparadigmen und -modelle zur bestmöglichen Nutzung architetkonischer Möglichkeiten von Prozessoren und Rechnersystemen
    - Für den Software-Entwuf existieren eine Vielzahl von Werkzeugen
  - Software-Ergonomie
    - Gesamtheit der Krieterien für eine Nutzerakzeptanz
    - Software für die Gestaltung der Beutzerschnittstelle ist oft umfangreicher als die Applikationssoftware
  - Algorithmen-Entwurf
    - Optimale Programme erfordern geeignete Lösungsalgorithmen
    - Besonders drastische Bedingungen bestehen bei Parallelverarbeitung durch erforderliche Parallelalgorithmen und Parallelisierung sequenzieller Algorithmen
    - Optimale Parallelalgorithmen können zum Einsatz sogenannter Systolischer Arrays führen

5.3.5 Entwurf eines Rechnersystems

- Kompromissfindung zwischen
  - Zielsetzungen: Anwendungsbereiche, Funtkionalität, Verfügbarkeit, ...
  - Gestaltungsgrundsätzen: Modularität, Sparksamkeit, Fehlertoleranz, ...
  - Randbedingungen: Technologie, Finanzen, Umwelt, ...
- Zielsetzungen
  - Andwendungsbereich
    - Technisch-wissenschaftlicher Bereich (z.B. Strömungsmechanik, Materialforschung, ...)
    - Kommerzieller Bereich (z.B.: Datenbankanwendungen, Internet, Suchmaschinen,...
    - Eingebettete Systeme (z.B.: Verarbeitung digitaler Medien, Automatisierungstechnik, ...)
  - Benutzerfreundlichkeit
    - Beziehung zwischen einem Rechnersystem und Nutzer
    - Gestaltung der Schnittstelle zwischen dem Rechnersystem und seinem Benutzer
  - Verlässlichkeit/Robustheit
    - Gewährleistung einer minimalen Verfügbarkeit des Systems
  - Erweiterbarkeit/Skalierbarkeit
    - Eine Rechnerfamilie ist in Ausbaustufen skalierbar für verschiedene Anwendungen
    - Skalierbarkeit ist ein wesentliches Erfordernis aller Rechnersysteme $\Rightarrow$ Chancen auf dem Markt
    - z.B. SGI Slogan: Paying by growing von Einstiegs-Servern bis zu HPC-Maschinen
    - Motivation: Architektur kennen und schätzen lernen $\Rightarrow$ Ideen für neue Projekte
    - Fragestellung: Welche architektonischen Voraussetzungen sind für die Erweiterbarkeit erforderlich?
  - Konsistenz
    - Eigenschaft des Systems mit folgerichtigem, schlüssigem Aufbau
    - Vorausschauender Entwurf einer Rechner- bzw. Prozessorarchitektur, der zu erwartenden Architekturerweiterungen schon Rechnung trägt
    - Beispiel: MIPS-Prozessor-Familie

  - Orthogonalität/Modularität
    * Funtkional unabhägige Teilelmente sind unabhängig voneinander spezifiziert und realisiert
    * Standardisierung hat immer größere Bedeutung
    * Hauseigene Lösungen ohne Zweitanbieter relativ chancenlos
    * Wichtiger Gestaltungsgrundsatz für Befehlssätze $\Rightarrow$ vereinfachte Code-Erzeugung in Compilern

- Gestaltungsgrundsätze
  - Symmetrie
  - Angemessenheit
  - Sparsamkeit
  - Wiederverwendbarkeit
  - Transparenz/Abstraktion
  - Virtualität

- Randbedingungen
  - Technologische
  - Finanzielle
  - Käuferakzeptanz
  - Moore's Law

5.3.6 Architectural Trends

- Architecture translates technology's gifts into performance and capability
- Resolves the tradeoff between parallelism and locality
- Understanding microprocessor architectural trends
- Greatest trend in VLSI generation is increase in parallelism
  - up to 1985: bit level parallelism: 4-bit $\rightarrow$ 8-bit $\rightarrow$ 16-bit
  - mid 80s to mid 90s: instruction level parallelism
  - End 90s: thread level parallelism and/or chip multiprocessors
  - Next step: reconfigurable computing
- How far will ILP go?
  - Infinite resources and fetch bandwidth, perferct branch prediction and renaming

- Thread Level Parallelism ön board"
  - Micro on a chip makes it natual to connect many to shared memory
- Problem: Prozessor-Memory Performance Gap: 50 percent per year

5.3.7 Bemerkungen zum klassischen Digitalrechner

- zentrale Steuerung durch Steuerwerk
- zentrale Verarbeitung durch Rechenwerk
- Harvard $\leftrightarrow$ von Neumann
- Harvard-Architektur
  - Vorteile
    - geringere Wartezeiten
    - einfache Busverwaltung
  - Nachteile
    - höherer Verdrahtungsaufwand (hohe Kabelkosten)
    - verminderte Flexibilität bei der Ausnutzung der Speicher (keine Austauschbarkeit)
  - Grundidee: Optimierung der Speicherhierarchie
- Von-Neumann-Architektur
  - Architektur des minimalen Hardware-Aufwands
  - Einzigartige Verbindung von
    - Einfachheit
    - Flexibilität
  - Grundelemente
    - Leitwerk und Rechenwerk
    - Hauptspeicher
    - Eingabeeinheit und Ausgabeeinheit
    - Verbindungseinrichtung
  - Vorteile
    - Einfachheit
    - maximale Flexibilität
  - Nachteile

* Befehle und Programmdaten müssen über einen Kanal zwischen Speicher und Prozessor transportiert werden (sog. physikalischer von-Neumann-Flaschenhals)
* streng sequentielle Abarbeitung (sog. intellektueller von-Neumann-Flaschenhals)
* große semantsiche Lücke zwischen den für die Benutzungsschnittstelle typischen höheren Programmiersprachen und den im von-Neumann-Speicher enthaltenen von-Neumann-Variablen

5.3.8 Aufgaben und Ziele der Rechnerarchitektur

Aufgaben

- Architekturanalyse bestehender Rechnersysteme und ihrer Komponenten, wie Prozessoren, Speicher, Verbindungseinrichtungen u.a.
- Beobachtung der Evolution von Rechnerfamilien und Architekturklassen sowie Ableitung neuer Architekturrichtungen
- Entwurf und Synthese neuer leistungsfähiger Rechensysteme mit bewährten Entwurfsmethoden und automatisierten Werkzeugen
- Umsetzung von Leistungsanforderungen, die von Anwendungsbereichen vorgegeben werden, in Struktur und Organisationsformen für Rechner und deren Komponenten

Ziele

- Leistungssteigerung durch Architekturverbesserungen
- Steigerung der Nutzerakzeptanz durch benutzergerechte System- und Anwendersoftware
- Entwurf ausbaufähiger Rechnerarchitekturen, die konkurrenzfähig bleiben und Weiterentwicklungen mit reduzierten Kosten gestatten

5.3.9 Klassifizierung nach Flynn

- SISD - single instruction stream, single data stream
- SIMD - single instruction stream, multiple data streams
- MISD - multiple instruction streams, single data stream
- MIMD - multiple instruction streams, multiple data streams

5.4 Intel Prozessoren

5.4.1 Ursprung

- 32-Bit Integer-Verarbeitungsbreite
- CISC Befehlssatz
  - Vairable Befehlslänge
  - Arithmetische Operationen mit Speicheradressen
  - 32-Bit Immediate Werte im Befehlswort
  - 1-Adress und 2-Adress Format
- SISD
- Virtueller Speicher

5.4.2 Intel 80386

IA-32 Register

- EIP: Instruction Pointer
- EFLAGS
  - Overflow-, sign-, zero- Flag
  - Für Verzweigungsbefehle
- 8 General Purpose Register
  - Häufig implizite Adressierung
- Segment Register zur Speicheradressierung

Stack

- Wegen geringer Registerzahl häufiges Ausweichen auf Speicher erforderlich
- Stack ist spezieller Speicherbereich in dem Werte zwischengespeichert werden

Adressierungsarten

- Immediate Operand
- Immediate Adresse
- Register Zugriff
- Register Indirekt
- Register indirekt mit displacement
- Zugriffe auf Stack (push/pop)
  - Register indirekt mit preautodecrement
  - Register indirekt mit postautoincrement
- Implizite Adressierung
  - Feste Quell- oer Zielregister für Befehle z.B.:
    - ein operand in EAX
    - Loop Counter in ECX
    - ESP/EBP/ESI/EDI für Adressberechnung
  - spart Bits im Befehl
- Speicher indirekt

Virtueller Speicher

- Mehrere Programme teilen sich den physischen Speicher
- Eigener virtueller Adressraum für jedes Programm
  - Adressierungsarten beziehen sich auf virtuelle Adressen
  - Übersetzung in physische Adressen erfolgt zusätzlich
- Ressourcenverwaltung durch Betriebssystem

5.4.3 Intel 80486

- Integrierter L1-Cache
- Integrierte FPU)486DX

Cache

- Kleiner, schneller Zwischenspeicher
- Puffer für häufig benötigte Daten und Befehle

Pipelining

- Aufteilung der Befehlsverarbeitung in 5 Phasen
  - Instruction Fetch (IF): Befehl laden
  - Decode1 (D1)
  - Decode2 (D2)
  - Execute (EX)
  - ...
- Mehrere Befehle gleichzeitig in Bearbeitung
- Speedup durch Pipelining
  - ermöglicht höhere Taktfrequenz
  - nach Anlaufphase wird ein Befehl pro Takt fertiggestellt

5.4.4 Pentium

- Getrennte L1-Caches; 8KB für Instruktionen; 8KB für Daten
- Zwei Befehle gleichzeitig in jeder Phase
  $\rightarrow$ Es werden zwei Befehle pro Takt fertiggestellt
- 2. Pipeline nur für einfache Befehle

5.4.5 Pentium MMX

- Erweiterung der ISA um SIMD Befehle
- 8 64-Bit Register
  - Abgebildet auf FPU Register
- Nur Integer Befehle

5.4.6 Pentium Pro

- CISC Befehlssatz hinderlich für effizientes Pipelining
  - Variable Ausführungszeit in Ausführungsphase (EX)
  - Direkte Verarbeitung von Speicheroperanden
    - Cache Misses blockeren nachfolgende Befehle
    - Steigende Speicherlatenz bei höheren Taktraten
  - $\Rightarrow$ Durchsatz (instruction per cycle) i.d.R. weit unter theoretischem Maximum
- Lösung:
  - RISC Verarbeitung in Rechenwerken
  - Out-of-order Execution zur Verdeckung von Speicherlatenzen

RISC Kern

- Übersetzung der x86 Befehle in internen RISC Befehlssatz (Microops)
- Eine Microop für simple Operationen
- Mehrere Microops für komplexe Operationen
  - Aufspaltung in Speicherzugriff und arithmetische Operation
  - Trigonometrische Funktionen als Mikroprogramm

Out-of-Order Execution

- Abarbeitung der Microops abweichend von der Programmreihenfolge
  - Scheduler (Reservation Station)
    - Sendet Microops an die Rechenwerke sobald Operanden verfügbar sind
    - Microops einer Instruktion müssen nicht zusammen abgearbeitet werden
  - Reorder Buffer (ROB)
    - Zusätzliche Register zur Speicherung der Ergebnisse (Register Renaming)
    - Zustand der für Software sichtbaren Register wird in Programmreihenfolge aktualisiert (In-order completion)

5.4.7 Pentium 2 / Pentium 3

- Weiterentwicklung des PPentium Pro
  - 3-fach superskalar
  - 2x 16KB L1-Cache
  - Zunächst Slot Prozessoren mit L2-Cache in zweitem Chip
  - später mit on-die L2-Cache wieder in Sockelbauweise
- Pentium 3 mit SSE

Streaming SIMD Extensions (SSE)

- Register FILL?
- Datentypen FILL?

5.4.8 Pentium 4

- Designziel hohe Taktfrequenz
  - Frequenz begrenzt durch Länge der Pipelinestufen
  - Extrem lange Pipeline durch wetieres Aufteilen der einzelnen Phasen
    - erste Modelle mit 20 Stufen, später 31 (vgl. Pentium 3: 10 Stufen)
  - 2006 eingestellt, da hohe Frequenzen u hohem Stromverbrauch führten
- Execution Trace Cache
  - Kein gewöhnlicher L1 Instruction Cache
  - Speicherung bereits dekodierter Instruktionen
- HyperThreading
  - Betriebssystem sieht zwei logische Prozessoren pro CPU
  - Doppelte Registersätze, gemeinsame Nutzung der Rechenwerke
- Befehlssatzerweiterungen: SSE2, ab 2004 SSE3

Netburst Mikroarchitektur

- Geringe Decoder Leistung
  - bei Misses im Trace Cache nur 1 Microop pro Takt
- 64 Bit breite SIMD Einheiten
  - SSE Befehle in 2 Operationen
- 126 Microops ROB
- 2 Integer ALUs
- 2 FP Einheiten
- L1 Cache pro Kern
  - 128 Bit lesen/schreiben pro Takt
- Hyperthreading
  - 2 Threads gleichzeitig

SSE Erweiterungen

- SSE 2
  - 64-Bit Floating Point Befehle (double precision)
  - 2 Operationen pro Befehl
- SSE 3
  - Mehrere Operanden aus einem Register
  - Reduziert Kopieroperationen

5.4.9 Pentium M/ Core Solo/ Core Duo

- Parallelentwicklung zu Pentium 4
- Basieren auf Pentium 3
- Energiesparende Architektur für Mobilrechner
  - Speedstep Technologie
    - Anpassung der Taktfrequenz an Auslastung
    - Reduziert Leistungsaufnahme während Programme ausgeführt werden

  - Clock Gating
    * Trennung von Logikblöcken vom Taktsignal
    * Reduziert Leistungsaufnahme in Leerlaufphasen

5.4.10 Intel64 Prozessoren (Auswahl)

- 2004 Pentium 3 /Pentium D
- 2006 Core2 Duo, Core2Quad, Xeon54xx
- 2008 Core i5/i7, Xeon 55xx (Nehalem)
- 2010 Core i3/i5/i7, Xeon 56xx (Westmere)
- 2011 2.Gen i3/5/7, Xeon E3 (Sandy Bridge)
  - integrierte GPU
  - Advanced Vector Extansions (AVX)
- 2012 3.Gen, Xeon E3 v2 (Ivy Bridge)
- 2013 4.Gen, Xeon E3 v3 (Haswell)

Befehlssatz

- IA-32 Befehlssatz auf 64 Bit Verarbetungsbreite erweitert
- Registeranzahl verdoppelt
  - Weniger Stackzugriffe nötig
  - Erhöhte Codegröße
- Größerer virtueller Adressraum

5.4.11 Multicore Prozessoren

- Moores Law
- Leistung einzelner Prozessoren nicht beliebig steigerbar
  - Höhere Grade an Superskalarität erhöhen Rechenleistung nicht proportional aufgrund von Abhängigkeiten zwischen Befehlen
  - Leistungssteigerungen durch Pipelining begrenzt durch Verzweigungen im Code
- Multi-Core Prozessoren
  - Leistungssteigerung durch Duplikation von Teilen des Prozessors
  - Erfordert Anpassung der Software an Nutzung mehrerer Kerne

Entwicklung des Energieverbrauchs

- Kühlung limitert Verbrauch pro Prozessor
- Mehr Kerne mit weniger Frequenz sind effizienter für parallele Anwendungen

5.4.12 Intel Core Microarchitektur Core2Duo

- Decodiert bis zu 4 Instruktionen pro Takt (5 mit Makroop-Fusion)
- 128 Bit breite SIMD Einheiten
- 96 Microops ROB
- Verarbeitung von bis zu 6 Microops gleichzeitig
- L1-Cache pro Kern
- L2-Cache und FSB von beiden Kernen genutzt

5.4.13 Intel Nehalem Mikroarchitektur

- Prozessorkerne kaum verändert gegenüber Core Mikroarchtektur
  - SSE 4.2
  - Hyperthreading
- L2-Cache pro Kern
- Gemeinsamer L4-Cache für alle Cores
- Integrierter Speichercontroller
- FSB ersetzt durch Intel QuickPath Interconnect

Kapitel 6

Übung

6.1 Einführung

6.1.1 von-Neumann

Komponenten des v. Neumann Architektur

- CPU
  - Steuerwerk
    - steuert die Befehlsabarbeitung
    - Befehlszähler (program counter)→ Instruction Fetch (Befehl holen) → Adresse des Befehls steht im pc
    - Befehlsregister → Befehlswort ins Befehlsregister laden
    - Befehlsdekoder → ID (Instruktction Decode)
    - zentrale Steuerschleife → EX (execute - Befehl ausführen)
      - CISC, Abarbeitung des Befehls unter Aufsicht der zentralen Steuerschleife
      - RISC → nutzt das Rechenwerk, ALU
    - Steuer -und Statusregister (Flag Overflow) → WB (Write Back)
  - Rechenwerk
- Speicher
  - Programmkode und Daten liegen im gleichen Speicher
- Bus
  - v.Neumannscher-Flaschenhals: Daten + Befehle müssen über den BUS
    - IF → Bus
    - ID

* EX → Bus (wenn Operanden geholt werden
* WB → Bus

6.1.2 v.Neumann vs. Harvard

Harvard

- Trennung von Befehls und Datenspeicher: Befehlsspeicher → VN → CPU → Verbindungseinrichtung (z.B. Bus (VN)) → Datenspeicher
- heutige Anwendung: Getrennter L1-Cache in L1I- und L1D-Cache

6.1.3 Def. von Brooks vs Giloi

Brooks (1962)

Rechnerarchitektur, wie andere Architekturen, ist die Kunst der Bestimmung von Nutzerbedürfnissen nach einer Struktur, die so zu entwerfen ist, dass sie jene Bedürfnisse so effektiv wie möglich hinsichtlich ökonomischer und technologischer Erfordernisse erfüllt.

- gilt auch für jede Bauarchitektur
- bis Ende der 70er Jahre bezog sich Rechnerarchitektur vor allem auf die Programmierschnittstelle
  - Maschinenbefehlssatz (meist Assemblerbefehle)
  - Interruptbehandlung (maskierbare + nichtmaskierbare Interrupts)
  - Registersatz
  - Adressierungsarten (Basisadressierung, indirekte Adressierung, direkte Adressierung)
  - Ein-/Ausgabe

Giloi

z.B. Maschinendarstellung eines Floating Point Wertes Single Precision

- Single Precision → 32 Bit
- IFEE 754: |Sign|Charakteristik (Exponent + Bias|Mantisse| → Mantisse wird so weit verschoben, bis führende 1 herausfällt

6.1.4 RA-Definition Begriffe

Rechnerarchitektur

- Hardware-Struktur
  - Hardwarebetriebsmittelstruktur
    - Prozessorstruktur
      - 1985 Intel 80 386 (erster 32-Bit Prozessor) → nur Integer Unit
      - 1987 Intel 80 387 (erster Floating Point Unit, FPU)
      - 1993 Pentium 1: V-Pipe(IU) und U-Pipe (IU oder Teil der FPU) → 2 Betriebsarten: IU+IU, IU+FPU
      - 1995 Pentium Pro: P6-Architektur → heutige Core-Architektur ist davon abgeleitet
    - Speicherstruktur
      - intern: Register (L1,L2,L3- Cache, DRAM, Festplatte, Archiv, ...)
      - zwischen Prozessoren: gemeinsamer Speicher ⇒ CPU 1 → CPU N haben gemeinsamen MEMORY ||
        verteilter Speicher ⇒ CPU 1, RAM 1 → CPU N, RAM N; verbunden durch Verbindungsnetzwerk
  - Verbindungsstruktur
    - intern
      - Adressbus
      - Steuerbus
      - Datenbus
    - extern
      - Verbindungsnetzwerk unterschiedlicher Typologie (Hypercube, 2D Gitter, ...)
  - Kooperationsregeln (z.B. Master-Slave)
- Operationspprinzip
  - Informationsstruktur
    - Klassen von Datentypen (Byte, Wort, ...)
    - Menge der Maschinendarstellungen der Datenobjekte
  - Steuerungsstruktur
    - Ablaufsteuerung, pc-getrieben → unsere üblichen Rechner
    - Ablaufsteuerung, datengetrieben (Datenflussrechner) → wenn die Daten da sind wird automatisch die Operation ausgeführt
    - Datenzugriffssteuerung → Zugriff über Adresslogik, einfache Wertzuordnung, Assoziativer Zugriff (Adresse und Inhalt werden gemeinsam gespeichert) → Caches

Begriffsklärung

RA /[HW-Struktur]/[HW-Betriebsmittel-Struktur]/[Prozessorstruktur]/[**Steuerwerk**]

RA /[HW-Struktur]/[HW-Betriebsmittel-Struktur]/[Speicherstruktur]/[**Register**]

RA /[HW-Struktur]/[Verbindungsstruktur]/[**Speicherbus**]

RA /[Operationsprinzip]/[Informationsstruktur]/[Menge der Maschinendarstellungen der Datenobjekte]/[**Festkommadatenformat** nach IEEE 754]

RA /[Operationsprinzip]/[Informationsstruktur]/[Klassen von Datentypen]/ Strukturdatentypen]/ [**Doppelt verkettete Liste**]

RA /[Hardware-Struktur]/[HW-Betriebsmittelstruktur]/[Speicherstruktur]/[**Cache**]

RA /[Operationsprinzip]/[Informationsstruktur]/[Menge der Maschinendarstellungen der Datenobjekte]/[**Gleitkomma-Datenformate** nach IEEE 754]

RA /[Operationsprinzip]/[Steuerungsstruktur]/Ablaufsteuerung]/ **Program-Counter getriebene Ablaufsteuerung**]

RA /[Operationsprinzip]/[Informationsstruktur]/[Menge der Funktionen, die auf die Datenobjekte anwendbar sind]/[**Assemblerbefehl: ADD R6, R4, R1**]

RA /[HW-Struktur]/[**Verbindungsnetzwerk zwischen den Prozessoren**]

RA /[Operationsprinzip]/[Steuerungsstruktur]/Dateizugriffsteuerung]/[**Zugriff auf den Cache**]

6.2 Einführung

6.2.1 Moores Law

Die Anzahl der Transistoren pro Chip verdoppelt sich alle 1,5 bis 2 Jahre (1965).

Was macht man mit diesen Transistoren?

- Erhöhung der Prozessorleistung
  → Taktfrequenz kann erhöht werden, da bei kleineren Transistoren kleinere Kapazitäten umgeladen werden müssen (CMOS-Technik)
  - 1985 Intel 80382 (erster 32-Bit Prozessor)
    → hatte nur eine Integer-Unit und keinen Cache
- Verarbeitungsbreite erhöht (4-bit, 8-bit, 16-bit, 32-bit, 64-bit)

- Erhöhung der Anzahl der Verarbeitungseinheiten
  - 1989: 80486
    * eine FPU (Floating Point Unit)
    * eine IU (Integer Unit)
  - Intel Itanium
    * 2 FPU's
    * 6 IU's

Weshalb wir die Anzahl der Verarbeitungseinheiten nicht weiter erhöht?

ILP (Instruction Level Parallelism)
Problem: Zu viele Funktionseinheiten führen zu keinem zusätzlichen Geschwindigkeitsgewinn (Sp. Speedup), da viele Befehle Datenabhängigkeiten aufweisen und so zu wenig unabhängige Befehle vorhanden sind.

Multicore-Prozessoren

2001 IBM: erster DualCore Prozessor

2005 Intel, AMD

Problem: Wenn man Leistung umsetzen will muss man sich mit paralleler Programmierung beschäftigen.

Lücke zwischen Prozessorleistung und Zugriffszeit auf den DRAM verkleinern

$\rightarrow$ Caches: benötigen ca. $\frac{1}{3}$ der Chipfläche

- L1-Cache: Harvard-Architektur
  - L1-Instruction-Cache
  - L1-Data-Cache
- L2-Cache: meistens nicht getrennt (es gibt Ausnahmen: Intel Itanium Motecito)
- L3-Cache gemeinsam

Integration weiterer Baugruppen

z.B.: GPU (Graphic Processing Unit)

6.2.2 Klassifikationen nach Flynn

a) Flynnsche Kategorien

(1) SISD Single Instruction Stream Single Data Stream

- klassischer von Neumann Rechner
- Single Core Processor unter bestimmten Voraussetzungen:
  - Eingang: sequentielle Folge von Befehlen
  - Ausgang: Sequentielle Folge von Ergebnissen

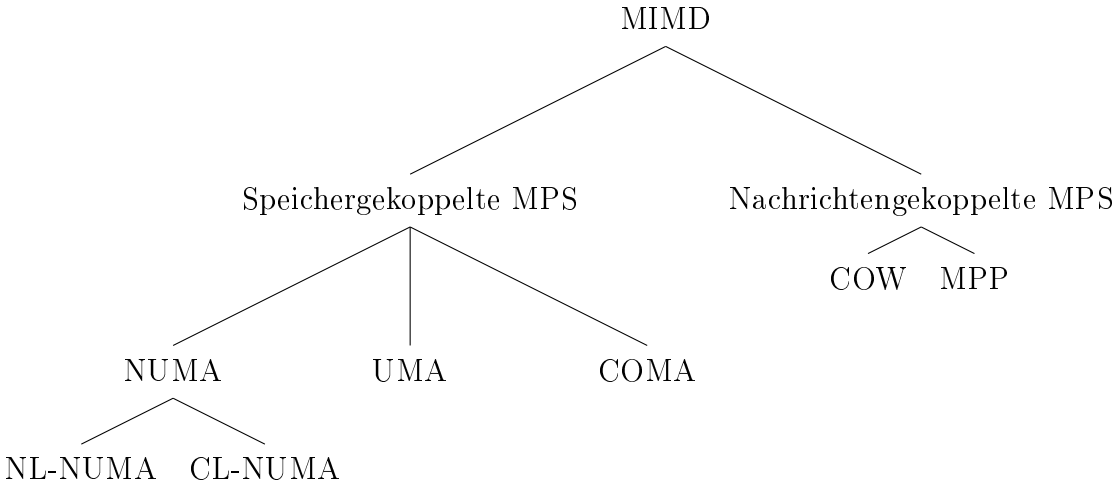
(2) SIMD Single Instruction Stream Multiple Data Streams

- Vektoraddition kann man durchführen mit
  - Vektorrechner
  - Feldrechner: ein Universalprozessor steuert die gesamte Abarbeitung, sehr viele einfache Verarbeitungseinheiten (processing elements, PE) führen zu einem Zeitpunkt die gleiche Operation aus.
  - Unterschiede:

(3) MISD Multiple Instruction Streams Single Data Stream
→ leere Klasse

(4) MIMD Multiple Instruction Streams Multiple Data Streams

- Multiprozessorsystem (MPS)
- Cluster von Workstations
- Nachteil: viel zu grob für alle MPS und COW
  → in der Literatur gibt es Erweiterungen zur Flynnschen Klasse MMID:



b) Flynssches Klassifikationsschema

Teil IV

Database

Kapitel 7

Vorlesung

7.1 Einführung

Gründe für DBS-Einsatz:

- Effizienz und Skalierbarkeit
- Fehlerbehandlung und Fehlertoleranz
- Mehrbenutzersynchronisation

ANSI - Database

- Standard siehe 1VL

Geschichte der Datenbanktechnologie

- siehe 1VL(28 ff.)

Databases vs Information Retrieval

- Information Retrieval 1VL(44)
  - Suche nach Dokumenten
  - Nimmt ständig zu
  - In welchem Datenbstadn wird gesucht? etc...

Databases vs Big Data

- Big Data 1VL(47)

7.2 Konzeptueller Entwurf

7.2.1 Drei Phasen des Datenbank-Entwurfs (4, ff.)

Phasen der SW-Entwicklung

- Anforderungs-analyse → Vorstudie
- Fachentwurf → Fachknozept
- IT-Entwurf → IT-Konzept
- Implementierung → Module/Klassen/DB-Tabellen

Phasen des DB-Entwurfs

- nach Fachentwurf: fachliche Anforderungen an Datenstrukturen → Konzeptueller DB-Entwurf → Konzeptuelles Schema (ER-D, UML, etc.)
- nach IT-Entwurf: Entscheidung für logisches (Implementierungs-)Modell → Logischer DB-Entwurf → Logisches Schema (relational, OO, etc.)
- nach Implementierung: Umsetzung in konkeretem System → Physischer DB-Entwurf → Physisches Schema (konkretes DBS)
- Datenbank = Schema + Daten

Datenbank = Schema + Daten

7.2.2 Lebenszyklus einer Datenbank

- Konzeptioneller Entwurf (12)
- Logischer Entwurf (13)
- Physischer Entwurf (14)
- Wartung, Modifikationen, Erweiterungen (14)
- Beispiel (15)

7.2.3 Prinzip eines Datenmodells (16)

- Grundlegendes Prinzip
- Leistung: Beschreibung
- Bestandteile
- Skizze (17)

7.2.4 Entity-Relationship-Modell

Entitäten (20)

- Definition
  - Existiert in der realen Welt, unterscheidet sich von anderen Entitäten
  - Eine Entität ist ein Objekt der realen oder der Vorstellungswelt, über das Informationen gespeichert werden sollen
  - Es ist im Sinne der Anwendung eindeutig berschreibbar und von anderen unterscheidbar
  - Gleichartige Entitäten werden zu Entitätstypen (Entitätsmengen) zusammengefasst
- Anmerkung
  - Welche Entitäten zusammengehören, ist von Semantik der Anwendung abhängig
- Merkmale von Entitätstypen (21)
  - Nur für die Anwendung relevante Merkmale werden modelliert
  - Beschreiben eine charakteristische Eigenschaft eines Eintitätstypes
  - Werte eines Attributes aus Wertebereichen wie INTEGER, REAL, STRING
- Schlüsselattribut(e)
  - Ein Attribut oder eine Menge von Attributen, anhand deren Entitäten eines Entitätstyps unterscheiden lassen
  - Werden durch Unterstreichung gekennzeichnet
  - Beispiel: die ISBN-Nummer identifiziert das Buch

Beziehungen / Relationships (22)

- Abbildung von Zusammenhängen zwischen Entitäten
- Homogene Menge von Beziehungen wird zu Beziehungstyp zusammengefasst
- binär / n-när
- Kardinalitäten Titel $\leftrightarrow$ Exemplar
- Bemerkungen
  - Ein Entitätstyp darf in einem Beziehungstyp mehrfach vorkommen
  - Mehr als zweistellige Beziehungstypen dürfen vorkommen
  - Beziehungstypen können auch Attribute besitzen

Beispiel eines ER-Diagramms (23)

Beispiel Funtkionalitäten (24)

Funktionalität von Beziehungstypen (25)

- Beispiele (26 ff.)

Besonderheiten (32 ff.)

- Rolle
  - Anfrage an DB: "Gib mir alle Angestellten, die mehr verdienen als ihr Chef"
- Extended-ER
  - Weak Enitites
    - ID nur im Kontext eindeutig (Bsp.: Stuhlnummer in Hörsaal 003 $\leftrightarrow$ Stuhlnummer in Hörsaal 004)
  - Strukturierte Attribute
    - Min-Max Beziehung (35 ff.)

Entwurf eines ER Diagramms (38 ff.)

Varianten für mehrstellige Beziehungstypen (40)

7.3 Konzeptueller Entwurf

7.3.1 Grundlagen (5)

Ausgangspunkt

- Idee und mathematische Formulierung geht zurück auf Ted Codd
- Basis für eine Vielzahl kommerzieller DBMS

Grundlagen

- Domänen / Wertebereiche: Integer, Sing[20], Datum, ...
- Relation R ist definiert auf einem Relationenschema RS:
  - RS: Menge von Attributen $\{A_1, \ldots, A_k\}$
  - Attribute: Wertebereiche $D_j = \text{dom}(A_j)$
  - Relation: Teilmenge des kartesischen Produkts der Wertebereiche $R \subseteq D_1 \times D_2 \times \ldots$

- weitere Terminologie (6)
  - Tupel: Element einer Relation
  - Kardinalität einer Relation: Anzahl der Tupel in einer Relation (endlich!!)
  - Darstellung der Relationen in Tabellenform
- Relationenschema (7): (Name, Einwohner, land) mit dom(Name) = String[40], ...
- Ausprägungen
- Allgemein

7.3.2 Primärschlüssel (8)

- Eindeutigkeit $\forall t_i, t_i \in R : t_i[x] = t_j[x] \Rightarrow i = j$
- Minimalität $\tau \exists Z \subset X(Z + X)$, so dass alle anderen Bedingungen gelten
- Definiertheit $\forall t_i \in R : t[x] \neq \text{NULL}$

7.3.3 ER-Modell Relationales Modell

Übersetzung von Entitäten

Eins-zu-Eins Übersetzung in Relationen

Übersetzung von Attributen

- Einfache Attribute
- Zusammengesetzte Attribute (z.B.: Adresse)
  - Berechnete Attribute (z.B.: Umsatz = Preis * Verkauf)
  - Mehrwertige Attribute (z.B.: Telefonnummer)

Übersetzung von Beziehungen

- Übersetzung von 1:1 Beziehungen (15)
  - Fall 1
  - Fall 2
  - Fall 3
- Übersetzung von 1:N Beziehungen (16)
  - Fall 1

  - Fall 2

- Übersetzung von N:M Beziehungen (18)
- Übersetzung von Beziehungen zwischen mehr als zwei Relationen
- Übersetzung rekursiver Beziehungen (19)
- Übersetzung von Attributen an Beziehungen (20)
- Übersetzung von Vererbungsbeziehungen

Abbildung der Vererbung 21

- Horizontale Partitionierung
- Vertikale Partitionierung
- Universalrelation (Typisierte Partitinierung)

Vergleich der Abbildungsvarianten

- Jedes Objekt genau ein Tupel in genau einer Relation, d.h. gleiche ID bedeutet nicht: dasselbe Objekt
- Gesamtheit aller Attribute eines Objekts nur durch Verbund zu ermitteln (teuer!)
- Referentielle Integrität unterstützt direkt Vererbung

7.4 Relationale Algebra

7.4.1 Motivation

- Formale Sprache, mit der sich Anfragen über einem relationalen Schmea formulieren lsassen
- Formale Sprache für den Berechnungsweg von Anfrageergebnissen
- Internrepräsentation für DB-Anfragen
- Mathematische Rechenregeln ermöglichen Abfrageoptimierung durch algebraische Umformung
- Nicht für den Nutzer eines DBMS sichtbar
- Auch geeignet zur Formulierung von Integritätsbedingungen

7.4.2 Relationale Algebra

- Gegeben eine Menge N(Anker der Algebra): Menge der Relationen
- Operationen opj: $N^k \rightarrow N$ (Abgeschlossenheit)
- 5 Basisoperationen

7.4.3 Basisoperationen

Projektion

Definition Sei A' eine Teilmenge der Attribute einer Relation $R(A_1, ...A_n)$. Die Projektion de rAttribute A' aus einem Tupel $t \in T$ ist definiert als das Tupel:

$$\pi_{A'} = (A'_1(t), \ldots, A'_m(t))$$

Die Projektion der Attribute A' einer Relation R ist definiert als die Relation

$$\pi_{A'}(R) = \{\pi_{A'}(t) | t \in T\}$$

heißt: Projektion ist eine Operation, die bestimmte Spalten aus einer Relation auswählt und diese als neue Relation ausgibt

- Da Dubletten (identische Tupel) in Relationen nicht vorkommen drüfen, enthält die Projektion i.A. weniger Tupel als die ursprüngliche Relation!
- ACHTUNG: Das ist in SQL standardmäßig nicht so!

Projektion in SQL

Projektion: $\pi_{\text{Name,Ort}}(\textit{Studenten})$
SQL Duplikate werden nicht standardmäßig eliminiert

```
1 SELECT Name, Ort
2 FROM Studenten
```

SQL mit Duplikat Eliminierung

```
1 SELECT DISTINKT Name, Ort
2 FROM Studenten
```

Selektion(Restriktion)

Definition Die Selektion einer Relation R ist definiert als die Menge aller Tupel aus R, die der Selektionsbedingung P genügen:

$$\sigma_F(R) = \{t | t \in R \wedge P(t)\}$$

P setzt sich zusammen aus:

- Operanden: Konstanten oder Name eines Attributs
- Vergleichsoperatoren
- Boolesche Operatoren

Selektion in SQL

$\sigma_{\text{Name = 'Schmidt'}}(Studenten)$

```
SELECT *
FROM Studenten
WHERE Name = 'Schmidt'
```

$\pi_{\text{Name, Vorname, Ort}}(\sigma_{\text{Name = 'Schmidt'}}(Studenten))$

```
SELECT Name, Vorname, Ort
FROM Studenten
WHERE Name = 'Schmidt'
```

Weitere Basisoperationen (15)

- Vereinigung $R \cup S$
- Differenz $R - S$
- Zusätzlicher Operator: Umbenennen von Relationen und Attributen: $p_S(R)$

Beispiele für Anfragen (18)

7.4.4 Abgeleitete Operationen

Durchschnitt und Division

- Durchschnitt:
  $R \cap S) := \{r | r \in R \text{ und } r \in S\}$
  Es gilt: $R \cap S = R - (R - S)$
- Division:
  $R \div S = \pi_{A-B}(R) - \pi_{A-B}((\pi_{A-B}(R) \times S) - R)$

Natürlicher Verbund

Natural Join: $\bowtie$

- Wichtigste Operation neben der Selektion
- **Definition** $R \bowtie S = \pi_{i_{k+1},\ldots,i_{r+s}}(\sigma_{R.A1=S.B1 \wedge \ldots \wedge R.Ak=S.Bk}(R \times S))$

Theta- und Equi-Join

- Theta-Join: $R \bowtie_\theta S$
  Definition $R \bowtie_{i\theta j} S = \sigma_{A_i \theta B_j}(R \times S)$ mit $\theta \in \{=, \neq, <, \leq, >, \geq\}$
- Equi-Join: Theta Join mit θ gleich '='

Verlustfreiheit von Joins

Definition Eine Join-Operation zwischen R und S heißt verlustfrei, wenn jeder Datensatz aus R und jeder Datensatz aus S in der Ergebnisrelation enthalten ist.

- Die inverse Operation Projektion erzeugt dann wieder R und S aus dem Join-Ergebnis
- Tupel, denen bei Join-Operationen die entsprechenden Tupel in der anderen Tabelle fehlen, mit denen sie verknüpft werden können heißen auch 'Dangling Tupel' bzw. Datensätze
- Um sie in die Ergebnismenge mit aufnehmen zu können, werden die Outer-Join-Operatoren benötigt
- Inner Joins sind in der Regel verlustbehaftet!

Outer Joins

- Left Outer Join: R ⟕ S
- Right Outer Join: R ⟖ S
- Full Outer Join: R ⟗ S

Kapitel 8

Übung

8.1 Einführung

8.1.1

a) Datenformat inflexibel:

   (1) keine stand. Pfade

   (2) viele Dateisys. erlauben keine Dateien über fixe Größe

   (3) keine stand. Datentypen (.txt) und Schema der Datenspericherung leicht uneinheitl.

b) mehrere Personen sollen/können darauf zugreifen

c) Datenformat Abfragesprache/-tool auch neu "lernenSSQL einheitl.

d) Redundanz sorgt für Anomalien (z.B. Aktualisierung d. Daten)

8.1.2

Rechteck: Entität, Raute: Bezeichner, Kreis: Attribute

a) 3

b) partielle Beziehungen

   - SxT→Ü
   - ÜxT→S
   - SxÜ→T

c) (1) Ein Tutor und ein Student nehmen an einer Übung teil

   (2) An einer Übung mit einem Tutor nimmt ein Student teil

(3) Ein Student in einer Übung hat einen Tutor

d) (1) und (3)

8.1.3

$$A : N, C : M, B : 1$$

Faustregel: Auf der rechten Seite steht eine 1.

8.1.4

ACHTUNG: Multiplizitäten genau andersrum wie bei 1.3

- T:(1,*)
- Ü:(1,*)
- S:(0,1)

8.1.5

- Bahnhöfe M ↔ 1 Städte
- Bahnhöfe 1 ↔verbindet↔ 1 Bahnhöfe
- Bahnhöfe 1 ↔verbindet↔ N Züge
- Bahnhöfe 1 ↔Start↔ L Züge
- Bahnhöfe 1 ↔Ziel↔ K Züge

8.2 ER-Modellierung

8.2.1 Prof-Stud

8.1

8.2.2 ER-Bahn

8.2

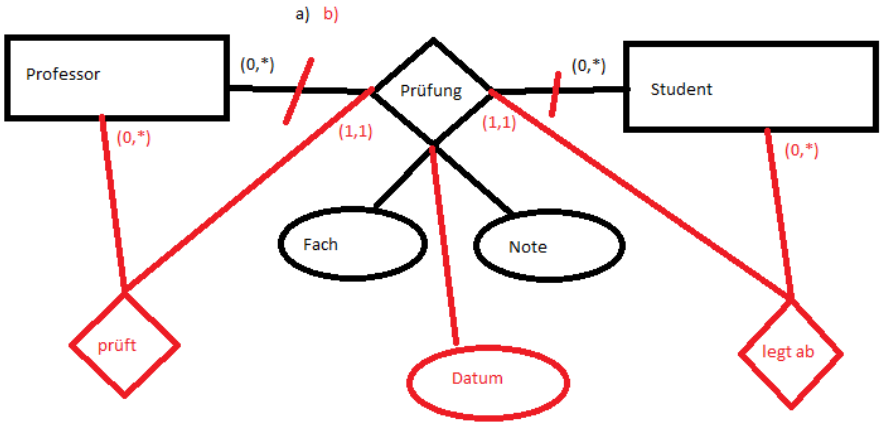


Abbildung 8.1: Professor-Student ER

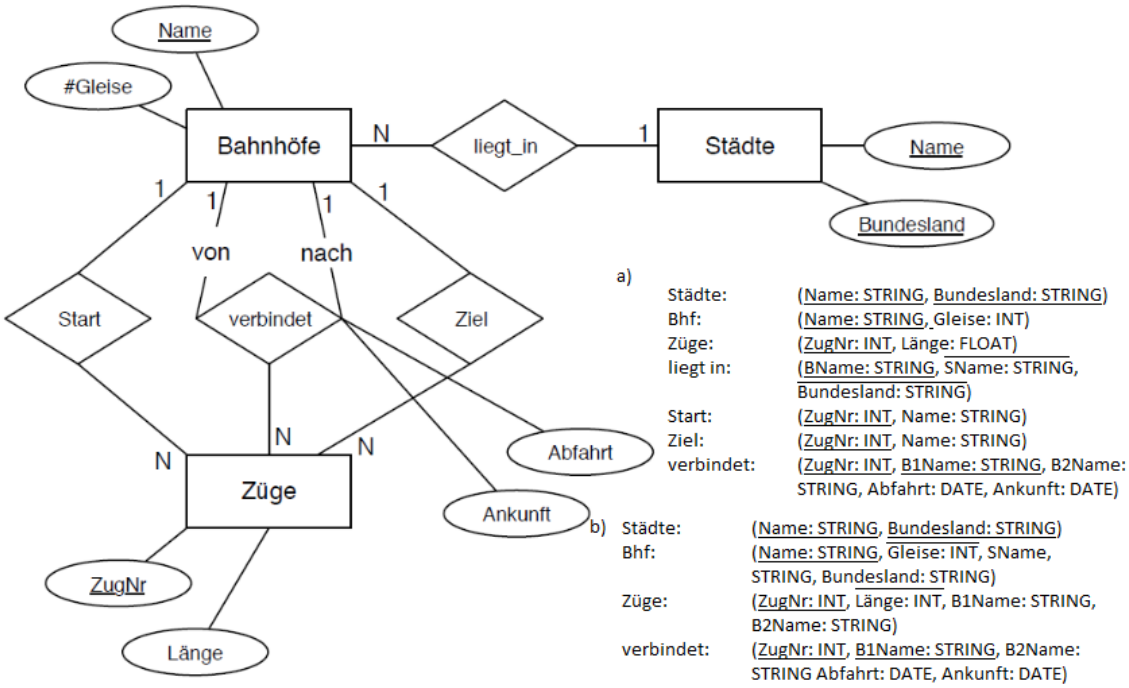


Abbildung 8.2: Bahnnetz ER

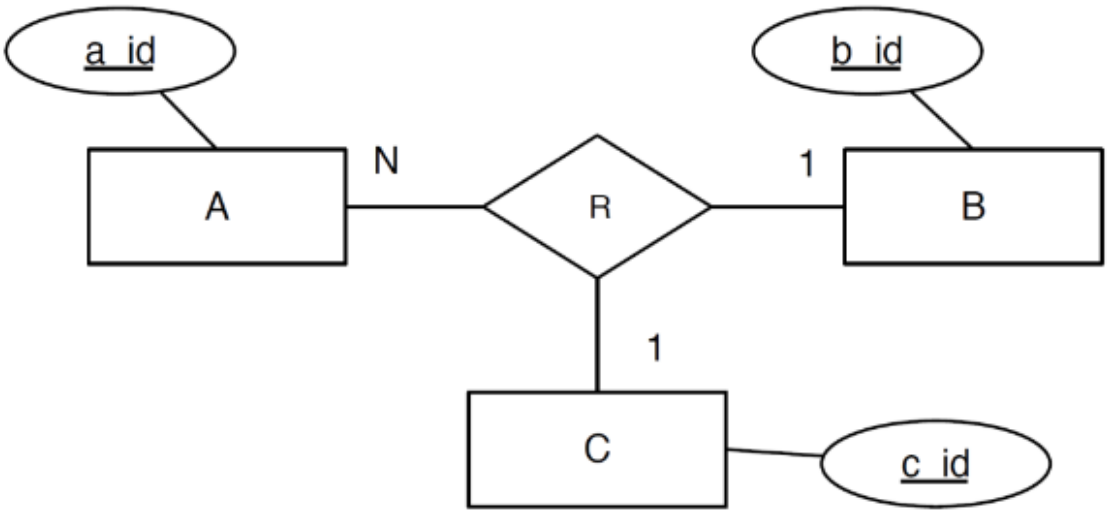


Abbildung 8.3: Relations ER

8.2.3 ER-Bsp

8.3

a)
- A x B → C
- A x C → B

b)
- A: (a id: INT)
- B: (b id: INT)
- C: (c id: INT)

c)
- R_1: (a id, b id, c id)
- R_2: (a id, b id, c id)

8.2.4 Vererbungshierarchie - Relationsschema

8.4

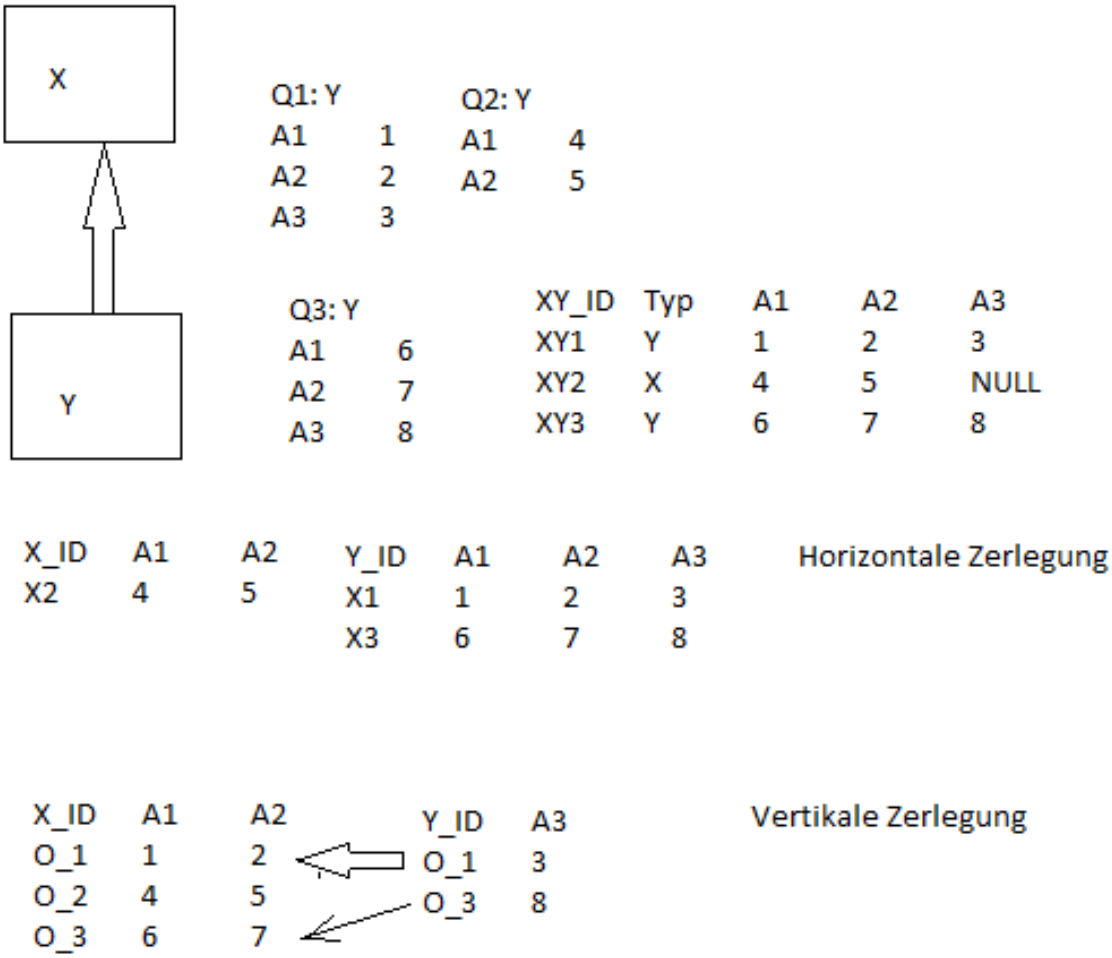


Abbildung 8.4: Relations ER

Teil V

Hardware Laboratory

Teil VI

C++4CG