SS 16

Luke Hain

23. Juni 2016

Inhaltsverzeichnis

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 → 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 → 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

1.4 Netztechnologien 2

1.4.1 Schichtenübersicht - Sicherungsschicht

1.4.2 Überblick

- Einordnung im Wesentlichen in OSI-Schichten 1 und 2
- PAN - Personal Area Network / LAN - Local Area Network
  - Ausdehnung bis zu einigen Kilometern
  - Privates Unternehmen / Privathaushalt als Netzbetreiber
- MAN - Metropolitan Ariea Network / WAN - Wide Area Network
  - weiträumige Ausdehnung, öffentlich zugänglich, dedizierte Betreiber
  - Dienstqualität sehr wichtig (viele konkurrierende Verkehrsströme
- hier: Technologien für MAN und WAN
  - MAN: Drahtloses Breitband, Token Bus/RPR, Carrier Ethernet
  - WAN: Carrier Ethernet, MPLS, SDA, OTN (3)

1.4.3 Drahtloses Breitband

WiMAX

- Problem: Kabel-gebundenes Internet sehr teuer, vor allem in ländlichen Gegenden
- Lösung: drahtlose Breitbandnetze
- Standardisierung: IEEE 802.16 = WiMAX (Worldwide Interoperability for Microwave Acces)
  - Frequenzbereich 2 GHz - 66 GHz
  - erste Version 2001: stationär, Sichtverbindung nötig
  - heute: für mobile Anwendungen mit Fahrzeuggeschwindigkeit geeignet
  - 4G-Technologie mit bis zu 1 GBit/s - verbindet Aspekte von WLAN und 3G
- Technologien
  - OFDM (Orthogonal Frequency-Division Multiplexing)
  - MIMO (Multiple Input Multiple Output)
- Prinzip, grafisch (5)

- Bitübertragungsschicht
  - OFDMA - Orthogonal Frequency Division Multiple Acces
    - flexible Aufteilung von OFDM-Unterträgern auf Stationen
  - Zeiitduplexverfahren TDD (Time Division Duplex)
    - Station wechselt zwischen Senden und Empfangen auf einem Unterträger (6)
- WiMax - Dienstklassen und Rahmenstruktur
  - Konstante Datenrate (z.B. unkomprimiertes VoIP)
  - Echtzeit mit variabler Datenrate (z.B. kompr. VoIP)
  - Nichtechtzeit mit variabler Datenrate - (z.B. Übertragung großer Dateien)
  - Best Effort
- Rahmenstruktur (7)
  - Anforderung Bandbreite
    Type → Bytes Needed → Connection ID → Header CRC
  - Datenübertragung
    Type → Flags → Length → Connection ID → Header CRC → Data → CRC

Steigerung der Datenrate: MIMO

- MIMO - Multiple Input Multiple Output
- Nutzung mehrerer Sende- und Empfangsantennen pro Gerät
- Signale überlagern sich, durch Mehrwege-Ausbreitung kleine Vraitionen im empfangenen Signal
- Kanalmatrix H wird beim Empfänger berechnet → sehr komplex
- bei n Sende- und n Empfangsantennen theoretisch n-fache Kanalkapazität möglich
- Komplexität der Berechnung steigt überproportional mit n
- Nachteil MIMO: höhere Hardware-Kosten

1.4.4 Token-basierte Technologien

Token Bus IEEE 802.4

- physikalischer Bus, logischer Ring (nur für Token-Weitergabe)
- Charakteristika: deterministisch, realzeitfähig, Prioritäten
- Vorläufer mit ähnlichen Eigenschaften: Token Ring (IEEE 802.5)
- Aufnahme neuer Stationen:
  - Token-Besitzer sendet periodisch Anfragen (SOLICIT-SUCCESSOR)
  - Station kann Äufnahmeantragßtellen
  - Kollisionsbehandlung, falls sich mehrere Stationen melden
- Verlassen des Ringes:
  - sende SET-SUCCESSOR an Vorgänger (mit Nachfolgerangabe)
  - Vorgänger setzt neuen Nachfolger
- Ausfall einer Station:
  - Vorgänger prüft nach Token-Weitergabe, b gesendet wird oder Token wiederum weitergegeben wird (ggf. mehrfach)
  - falls nicht, sende WHO-FOLLOWS, übernächste Station wird dann durch SET-SUCCESSOR zum neuen Nachfolger
  - Ausfall des Token-Besitzer: Neuinitialisierung
- Mehrfach-Token (durch zufällige Duplizierung etc.):
  - Erkennung bei unberechtigten Sendevorgängen
  - freiwilliges Löschen des eigenen Tokens
- Zahlreiche weitere Verwaltungsprotokolle

RPR - Resilient Packet Ring

- RPR: Weiterentwicklung von Token Ring / Token Bus für den MAN/WAN Bereich (IEEE 802.17)
- Datenraten: 1 GBit/s bis 10 GBit/s
- doppelter Ring (Ringlets) für beide Richtungen mit optischen Verbindungen
- Dienstklassen: real time, near real time, best effort

- Fairness: koordinierte Drosselung aller Datenströme bei Überlastung
- wichtigstes Feature: schnelle umschaltung auf Ersatzwege bei Leitungsunterbrechungen (in 50 ms)
- Spatial reuse: mehrere gleichzeitige Übertragungsvorgänge pro Ringlet möglich → bessere Ausnutzung der Übertragungskapazität

1.4.5 Carrier Ethernet

- Ethernet-Technologie hat sich auch im MAN/WAN Bereich durchgesetzt und dort andere Technologien (ATM, FDDI, Fibre Channel) verdrängt
- Gründe: niedrige Kosten, kompatibel zu LAN-Paketen, flexibel, einfach zu nutzen
- Voraussetzung: Punkt-zu-Punkt-Verbindungen (Connection-oriented Ethernet)
  - kein CSMA/CD
  - Wegfall der Längenbecshränkungen
- mit Glasfaserkabeln heute Ethernet-Verbindungen von 10 bis 140 km bei 1 bis 100 GBit/s realisierbar
- weitere Optimierungen (siehe auch Kap. 8):
  - Jumborahmen bis 9KB
  - Bitübertragung durch 8B/10B-Codierung bzw. 64B/66B-Codierung mit geringem Overhead

Metro Ethernet Networks MEN

- Ethernet-basierte Dienste im MAN/WAN-Bereich
- Standardisierung: Methro Thernet Forum (MEF) und IEEE
- MEF-Standards enthalten Ethernet-Erweiterungen zu
  - DIenste:E-line, E-LAN oder E-Tree
  - Skalierbarkeit: inkrementell skalierbare Datenrate
  - Zuverlässigkeit: Unterbrechungen entdecken und schnell beheben (50ms)
  - qquality of Service: Service Level Agreements (SLAs)
  - Service Management: Moitoring, zentrale Steuerung
- verschiedene Transporttechnologien möglich: MPLS, SDH, OTN
- Beispiel (14)

Provider Bridging

- Provider Bridges - IEEE 802.1as QinQ (Stacked VLANs)
- Erwetierung des Ethernet Frames:(15)
- Outer Tag und Destination Address werden zum Routing im Provider-Netz verwendet
- Weiterentwicklung: Provider Backbonde Bridges - 802.1ah Mac-in-Mac
- neue Ethernet-Header: (15)

Dienstgüte bei Carrier Ethernet

- MEF spezifiziert Bandwidth Profiles für Ehternet Virtual Connection (EVC) = Service Level Agreement (SLA)
  - Ingress Badnwidth Profile - Spezifikation für eingehenden Traffic eines User Network Interface (UNI)
  - Egress Bandwidth Profile - Spezifikationen für ausgehenden Traffic an ein UNI
  - Commited Information Rate (CIR) - feste zusage für kritische Daten
  - Excess Information Rate (EIR) - weitere Kapazität für unkritische Daten
- Traffic Coloring
  - Grün: Service Frame liegt innerhalb der SLA = CIR
  - Gelb: Sercie Frame liegt außerhalb der SLA, wird aber auf Best Effort Basis weitergeleitet = EIR
  - Rot: Service Frame wird verworfen
- Auch Beschränkungen für Delay, Paketverluste und Jitter können Bestandteil des SLA sein

Exkurs: VPNs

- VPN = Virtual Private Network
- Motivation
  - Anbindung entfernte Bürostandorte
  - Einbindung mobiler Mitarbeiter
  - Einrichten eines Forschungsnetzes (Bsp.: CERN)
- Funktion

  - Einrichten dedizierter privater Kommunkationpfade ("Tunnel") durch ein öffentliches Weitverkehrsnetz
  - meist kombiniert mit Verschlüsselung für Abhörsicherheit zischen zwei Zugangspunkten

- Realisierung
  - Layer 3 (Vermittlungsschicht): z.B. per IPsec - IP-basierter Tunnel zwischen 2 Endpunkten (Client und VPN-Gateway) oder 2 Routern
  - Layer 2 (Netzwerkschicht):
    * Einrichten virtueller LANs (z.B. Carrier Ethernet)
    * Einrichten virtueller Pfade durch ein WAN (z.B. Multi Protocol Label Switching - MPLS)

1.4.6 MPLS - Multiprotocol Label Swtitching

- Motivation
  - Schnelle Weiterleitung, QoS-Garantien
  - Entlastung von Routern
  - Trennung von Netzwerktraffic (z.B. VPNs, QoS-Klassen)

- Merkmale
  - Weiterleitung von Paketen entlang vordefinierter Pfade (gesteueret druch Labels)
  - Forward Equivalence Classes (FEC), d.h. gleiche Behandlung aller Pakete eines Datenstroms (z.B. Videokonferenz)
  - Abbildung auf existierende Netztechnologien möglich bzw. Implementierung durch spezielle MPLS-Hardware direkt über Lichtwellenleiter

- MPLS zusammen mit Carrier Ethernet
  - Variante 1: MPLS-Router per Carrier Ethernet verbunden → MPLS over Carrier Ethernet
  - Variante 2: Carrier Ethernet als Overlay über optisches MPLS-Netz realisiert → Carrier Ethernet over MPLS

- Funktionsweise/Beispiel (19ff)

MPLS-Switches / Layer-3-Switches

- Kombination von Router und Switch
- zunächst Routing, bei längerer Dauer von Datenströmen ggf. Druchstalten von Verbindungen auf Basis von MPLS (Multiprotocol Label Switching)
- Teilweise auch Berücksichtigung von Informationen der Transportschicht ("Layer-4-Switches")

MPLS Transprot Profile (MPLS-TP)

- verbindungsorientiertes MPLS, für Transprotnetze (Backbone) optimiert
- durch IETF standardisiert, Nachfolger von T-MPLS (ITU-Standard)
- vordefinierte Routen - bidirektional
- QoS-Erweiterungen, Netzwerkmanagement
- weitere Funktionen für Überwachung und Verwaltung des Netzes (in-band)
- Unterstützung für Netzwerkmanagement-System (NMS)

MPLS - Bewertung

- gute Integration mit IP und Ethernet, einfaches Netzmanagement
- es wird ein eigener Header für MPLS eingeführt /zwischen IP-Header und Header des darunterliegenden Ethernet bzw. anderer Netze); dadurch wird Unabhängigkeit vom Basisnetz erreicht ("Multiprotocol")
- mehrere Anwendungen können zu einer Forward Equivalence Class zusammengefasst werden; daher flexibel und ressourcenschonend
- MPLS-Pfade müssen nicht explizit aufgebaut werden, sonder werden ön-the-flyëtabliert, z.B. durch Analyse des Datenstromes
- in der Praxis heute häufig (10/100) Gigabit Ethernet auf Basis von Hochleistungsswitches kombiniert mit MPLS im WAN-Bereich im Einsatz

1.4.7 SONET / SDH

- SONET - Synchronous Optical Network, SDH - Syhnchronous Digital Hierarchy
  - Bündelung mehrerer Datenströme über optische Leiter
  - wird im Telefonie-Backbone welweit eingesetzt

- ISDN-Telefonie: 8000 · 8 Bit/Sample = 64 kBit/s
  - Idee: jede Telefonverbindung belegt ein Byte pro SONET/SDH-Rahmen
  - Rahmen alle 125 μs senden
- SONET Basisrahmen: Block von 810 Byte = STS-1
  - 8 · 810 Bit · 8000 · 1/s = 51,84 MBit/s
  - 9 Zeilen und 90 Spalten - 3 Spalten Overhead, 87 Spalten Nutzdaten
- höhere Datenraten durch größere Pakete (Vielfache von STS-1)
- hierarchisches Multiplexing/Demultiplexing

1.4.8 OTN - Optical Transport Network

- Problem SONET/SDH: keine höhere Geschwindigkeit als 40 GBit/s
- OTN - einheitliche Übertragungstechnologie für optische Netze
  - Nachfolge-Technologie von SONET/SDH
  - kann SONET/SDH-Traffic transportieren
  - kann auch anderen Traffic wie Ethernet, ATM oder MPLS gemsicht über eine optische Verbindung übertragen
- stadardisiert durch die ITU (G.709)
- Datenübertragung basiert auf DWDM - Dense Wavelength Division Multiplexing
- unterstützt Datenraten größer als 40 GBit/s
- verbesserte Zuverlässigkeit
- Funktionen
  - Forward Error Correction
  - Frame Mapping
  - Multiplexing (DWDM)
  - Management von Pfaden und Performance
- Beispiel (27)

DWDM - Dense Wavelength Division Multiplexing

- WDM - Wavelength Division Multiplexing
  - Aufteilung der Kanäle auf verschiedene Wellenlängen
  - Hinzufügen/Trennung und Verstärkung durch rein optische Elemente möglich
  - durch bessere Sende-Empfangstechnik können bestehende Glasfasern höhere Datenraten transprotieren · Kosteneffizienz
- CWDM - Coars Wavelength Division Multiplexing
  - breite Trennbereiche zwischen Kanälen, ca. 5 bis 20 Kanäle pro LWL
  - kostengünstige Sende-/Empfangstechnik
- DWDM - Dense Wavelength Division Multiplexing
  - kleinere Trennbereiche · aufwendigere Sende-/Empfangstechnik
  - ca. 60 - 160 Kanäle pro LWL
  - Reichweite: ca. 80 - 160 km bei 100 Gbit/s; bis zu 1 TBit/s theoretisch möglich

1.5 Sicherungsschicht

1.5.1 Sicherungsschicht

Aufgaben

Kommunikation zwischen Partnern im gleichen Subnetz (z.B. via Ethernet) bzw. über Punkt-zu-Punkt-Verbindung (z.B. PPP - Point-to-Point-Protocol bei Internet-Zugang oder via VPN - Virtuelles Privates Netz)

- Bildung von Übertragungsrahmen zur Kontrolle von Prüfsummen
- Fehlerbehandlung
- Flusskontrolle zur Überlastvermeidung
- Verbindungsverwaltung
- Interface für die Vermittlungsschicht

Zuverlässigketisebenen

- Unbestätigter verbindungsloser Dienst
- Bestätigter verbindungsloser Dienst
- Bestätigter verbindungsorientierter Dienst
  - Aufbau einer Verbindung vor der Übertragung
  - nummerierte Rahmen
  - Garantie, dass der Rahmen ankommt
  - Reihenfolge wird beibehalten
  - genau einmaliger Empfang möglich

Rahmenbildung

Realisierung durch:

- Bytezahl am Anfang des Rahmens
- Rahmenbegrenzer auf Sicherungssschicht (Flagbytes, Flagbits)
- Rahmenbegrenzer auf der Bitübertragungsschicht
- kombinierte Verfahren, z.B. Ethernet und WiFi: Präambel als Anfang des Rahmens + Bytezahl für die Länge des Rahmens

Problem bei Rahmenbegrenzern: Bitmuster des Rahmenbegrenzers muss auch im Payload zulässig sein: Byte-/Bit-Stuffing

Bytezahl (6)

Bitstuffing

Ablauf: Füge nach jeder fünften aufeinanderfolgenden 1 in den Nutzdaten jeweils eine 0 ein und entferne diese wieder beim Empfänger → eindeutige Rahmenbegrenzung (Beispiel (7))

1.5.2 Fehlerbehandlung

Probleme: Verfälschungen durch thermisches Rauschen, Impulsstärungen, frequenzabhängige Verzerrung etc. → Einzelbitfehler und Bündelfehler
Hamming Distanz d: Minimale Anzahl unterschiedlicher Bits zwischen zwei Quellcodewörtern; hier d = 3 Anzahl erkennbarer Fehler: d-1, Anzahl korrigierbarer Fehler: (d-1)/2

Fehlererkennende Codes

Paritätsicherung (9)

Cydlic Redundancy Check - CRC

Funktionsweise und Beispiel (10 ff)

1.5.3 Protokolle

Stop-and-Wait-Protokoll(14)

Protokoll mit Fehlerbehandlung

Ablauf:

- Wiederhole verlorengegangene nachrichten nach Timeout, sonst sende nächsten Frame
- Ignoriere entstende Duplikate → Laufnummern 0/1 Erkennung

1.5.4 Schnittstellenereignistypen

Primitive des Dienstes

- Dienstname
- Dienstpritivname
- Grundform der Dienstleistung

Dienstbeschreibung durch Zustandsdiagramme (18ff)

Protokollbeschreibung (23ff)

Point-to-Point Protocol PPP

- Protokoll der Sicherungsschicht für Interet-Zugang, z.B. über Modem, ISDN, xDSL, ...
- Protokoll der sicherungsschicht für Packet over SONET und ADSL
- Network Control Protocol NCP
- Link Control Protocol LCP
- Funktionsweise basierend auf HDLC
- Ablauf (29)

ADSL

- Überblick und Protokollstapel: verbindet Privathaushalte über bestehende Telefonleitung mit dem Internet
- ATM (Asynchronous Transfer Mode)
- AAL5
- AAL Rahmen (31)

1.6

1.7

1.8 Netzwerkperformance

Sicherungs- → Transportschicht

1.8.1 Einführung

Leinstungsprobleme in Rechnernetzen

- Überlastung von Routern
- Ungleichgewicht der strukturellen Ressourcen
- synchrone Überlastung
- mangelhafte Systemabstimmung
- Jitter

1.8.2 Messung der Netzwerkleistung

- Möglichkeiten: Timer setzen oder Zähler für Häufigkeiten
- zu beachten:
  - Samplegröße groß wählen → Durchschnittswert
  - repräsentative Samples
  - Caching/Puffer deaktiviert
  - ohne äußere Einflüsse
  - Verwendung einer präzisen Computeruhr
  - vorsichtiges Extrapolieren von Ergebnissen

Leistungsoptimierung

- viele verschiedene Mechanismen, Techniken und Kofigurationsmöglichkeiten
- für Performance-Vergleiche ist Optimierungsziel festzulegen, z.B.
  - minimale Paketübertragungszeit
  - max. Gesamtnetzdurchsatz
  - Durchsatz und Übertragungszeit nur für bestimmte Datenklassen und/oder Sender optimieren
  - minimale Verlustrate
  - minimaler Jitter
  - minimale Paketvertauschungen
- bei Optimierung müssen meist mehrere Schichten gleichzeitig betrachtet werden → Wechselwirkung von Maßnahmen

Performaceverlust - Beispiel

siehe S. 7

1.8.3 Leistungsaspekte Ehernet

Leistungssteigerung von 10 MBit/s zu 100 GBit/s

- Verkürzung maimaler Kabellängen
- Erhöhen der minimalen Framelänge - Auffülen kurzer Frames (Padding)
- Paketaggregation - mehrere kleine Frames in einem Frame übertragen (Bursting)
- 4B/5B-Codierung oder 8B/10B-Codierung statt Manchester-Codierung
- höhere Taktrate/Symbolrate
- Jumborahmen

Jumbo-Frames

- proprietäre Ethernet-Erweiterung
- Vorteile
  - TCP-Lastkontrolle: schnellerer Anstieg der Datenrate nach Paketverlust
  - weiniger Header-Overhead - 5,7% bei 1500 Byte, 1% bei 9000 Byte → 4 - 5 % mehr Datendurchsatz

  - reduzierte CPU-Last (weniger Pakete zu verarbeiten, weniger Routing-Entscheidungen, etc.)

- Nachteile
  - nur effizient einsetzbar, wenn Ende-zu-Ende-Unterstützung
  - CRC-32 nicht robust genug bei großen Frames (>12KB)
  - Delay und Jitter werden leicht erhöht

Jumbo-Frames und TCP-Lastkontrolle

siehe S.10

Padding/Bursting: Motivation

siehe S.11

Afubau eines Ehernet Frames

siehe S.12

Berechnung Worst-Case-Effizienz

sihe S.13

1.8.4 Fairness

- Trotz hoher Leistungsanforderungen muss Fairness gewährleistet sein
  - Alle Teilnehmer greifen gleichberechtigt und gleichmäßig auf das Netzwerk zu
- Beispiel: Transatlantisches Telefonkabel Nr. 14(TAT-14)
  - Finanziert durch Konsortium aus mehreren Providern (z.B. Deutsche Telekom, Concert, France Telecom, Sprint)
  - Fairer Zugriff auf vorhandene Ressourcen für alle Provider

Fairness - Algorithmen

siehe S. 15 ff.

- FIFO, FCS, Taildrop
- Round-Robin-Fair-Queueing
- Gewichtetes Fair Queueing

- Prioritätsscheduling
- Scheduling anhand von Paketzeitstempel

Fairness - Forschung

- Vergleichsmessungen verschiedener Algorithmen
  - First-Come, First-Served (FCFS)
  - Elastic Round Robin (ERR): Round-Robin-Erweiterung mit besseren Fairness- und Leistungseigenschaften
- Offene Forschungsfragen gemäß "The Fairness Challenge in Computer Networks"
  - Erweiterungen für Fairness in Multicast- und Broadcast-Szenarien
  - Erweiterung auf andere QoS-Parameter: Faire Verwaltung und Steuerung von Paketverzögerung, Jitter, Verlustrate

1.8.5 Überlastungsüberwachung

Gewünschte Bandbreitenzuordnung

- Problem: zu viele Pakete werden ins Netz geschickt → Überlastung des Netzes
- Vermeidung: Überwachung der Überlastung auf Schichten 3 und 4
- optimiertes Ziel: Bandbreite optimal auf Transportinstantzen im Netz verteilen

1. Effiziente Senderate
   - falsche Annahme: 100 MBit/s- Verbindung auf 4 Transportinstanzen zu je 25 MBit/s verteilen
     * knapp unterhalb der Kapazitätsgrenze (<100 MBit/s) treten bei kurzzeitigen Paketschüben Pufferverluste auf
     * Verzögerung steigt durch Neuübertragung von Paketen → führt zu noch mehr Last → exponentieller Anstieg
   - Lösung: Bandbreite nur zordnen bis Verzögerung stark ansteigt
2. Max-Min-Fairness. Rate steigt für alle Datenflüsse gleichmäßig an. Wenn Engpass erreicht werd, wachsen nur restliche Datenflüsse weiter. (Beispiel für faire zuteilung: Kanäle mit je 300 MBit/s, 4 Datenflüsse A-D)
3. Konvergenz
   - Problem: stark schwankende Bandbreitennachfrage im Netz
   - Algorithmus zur Überlastungsüberwachung muss möglichst schnell gegen faire und effiziente Zuordnung konvergieren

Regulierung der Senderate

- beeinflusst von
  1. Flusskontrolle (Empfänger hat zu wenig Puffer) → Regulierung über Fenster mit variabler Größe
  2. Überlstungsüberwachung (geringe Kapazität des Netzes) → auf Feedback vom Empfänger reagieren
- Beispiele Überlastungsüberwachung
  - TCP klassisch - Paketverlust führt zu "Neustart"(Slow Start) der Senderate, Verdopplung bis Schwellwert
  - Erweiterung Fast-Retransmit - bei fehlenden Paketen werden Folgepakete doppelt vom Empfänger bestätigt (Dup-Acks) → Sender erkennt Dup-Acks und sendet fehlendes Paket vor Ablauf des Timers für Paketverlust
  - Erweiterung TCP-Reno - Fast Recovery: bei Empfang von 3 Dup-Acks (frühes Zeichen für Paketverlust) wird Congestion Window halbiert (vermeidet Slow Start)
  - Fast TCP - Übertragungsverzögerung (Ende-zu-Ende) wird gemessen und Datenrate wird angepasst

Fast TCP

- Protokoll mit neuem TCP congestion avoidance Algorithmus für schnelle Netze mit langen Verzögerungen (Long Fat Networks)
- Motivation: TCP-Nachteile:
  - geringe Paketverluste sind Voraussetzung für einen kontinuierlich hohen Datenstrom
- Gegüber TCP verwendet FAST TCP die Verzögerung des Netzwerks als Feedback um die Senderate anzupassen → Datenrate bleibt konstant
  - RTT (round trip time) - Tatsächliche Zeit der Paketübermittlung
  - base RTT - Zeit der Paketübermittlung ohne Warteschlangen
  - Unterschied beider Werte gibt angenommene Paketanzahl in der Warteschlange an
  - viele Pakete in der Warteschlange → Senderate wird reduziert
  - wenige pakete in der Warteschlange → Senderate wird erhöht

TCP-Reno vs. Fast TCP

sihe S.22

1.8.6 Leistungsteigerung

Hostdesign

- System kann durch Netzwerkoptimierungen nur bis zu einem gewiisen Punkt verbessert werden → zugrunde ligendes Hostdesign optimieren
- Hostgeschwindigkeit (CPU + Speicher) wichtiger als Netzgeschwindigkeit
- weniger Overhead durch groß gewählte Pakete
- weniger Kopiervorgänge der Daten zwischen Schichten
  - Zusammenfassen der Schichten
  - parallele Operationen
- weniger Kontextwechsel
- weniger Timeouts, da unnötige Wiederholungen die Hostressourcen belasten
- grundsätzlich: Überlastung vermeiden ist besser als Überlastung beheben

Protokolle für den Normalfall optimieren

s.S. 25

Engpass: Beschränkte Bandbreite

s.S. 26

Long Fat Networks

s.S. 27

Bandwidth-Delay-Product

s.S. 28

Lösungen für Long Fat networks

s.S. 29

1.8.7 Unterberungstolerante Netzwerke - DTN

s.S. 30 fff

DTN - disruption-tolerant network

s.S. 30

1.9 Internetdienste

1.9.1 Domain Name System

- Abbildung von Namen auf IP-Adressen
- hierarchisches, auf Domänen basierendes Benennungsschema
- replizierte Verzeichnisdatenbanken
- Namensvergabe durch ICANN (Internet Corporation for Assigned Names and Numbers) - bzw. durch beauftrage Unternehmen

Resource Records

- zu jeder Domäne gehören entsprechende Resource Records
- typischer Resource Record ist Zuordnung Host → IP-Adresse
- Aufbau: <Domain_name, Time_to_live, Class, Type, Value>
- Beispiel: stargate 86400 IN A 141.76.40.230
- Erläuterung s.S. 5

Nameserver (6)

Namensauflösung für entfernte Domänen (7)

1.9.2 E-Mail (8)

Nachrichtenformat - RFC 5322 (9)

MIME (10)

SMTP (11ff)

IMAP (13)

WWW (14)

1.9.3 WWW - World Wide Web

Ablauf Abruf einer Webseite (15)

Webserver (16)

HTTP(17ff)

HTML (20ff)

Dynamische Webseiten (24ff)

1.9.4 Content Delivery (25)

Serverfarmen (26)

Webproxys (27)

Content-Delivery-Netze (28)

Peer-to-Peer-Netze (29)

1.9.5 Netzwerkmanagement (30 ff)

SNMP - Simple Network Management Protocol (22)

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 $\rightarrow$ Tiefpass (f_g) $\rightarrow$ Abtastung (zeitdiskret, (f_a)) $\rightarrow$ 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)

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

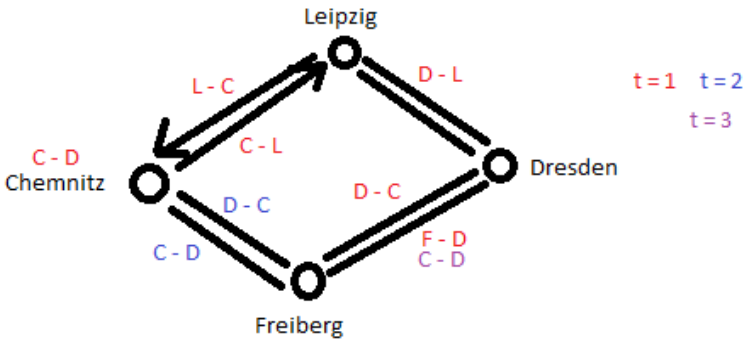


Abbildung 2.1: RPR a) und b)

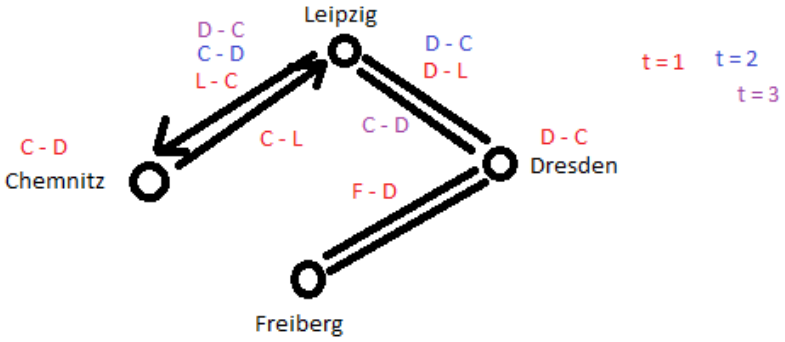


Abbildung 2.2: RPR c)

2.4 Netztechnologien 2

2.4.1 WiMax

siehe Musterlösung

2.4.2 RPR

a) Zeichnen Sie eine Topologie mit einem RPR Router pro Universität und den entsprechenden Ringlet-Verbindungen

b) Abb.: 2.1

c) Abb.: 2.2

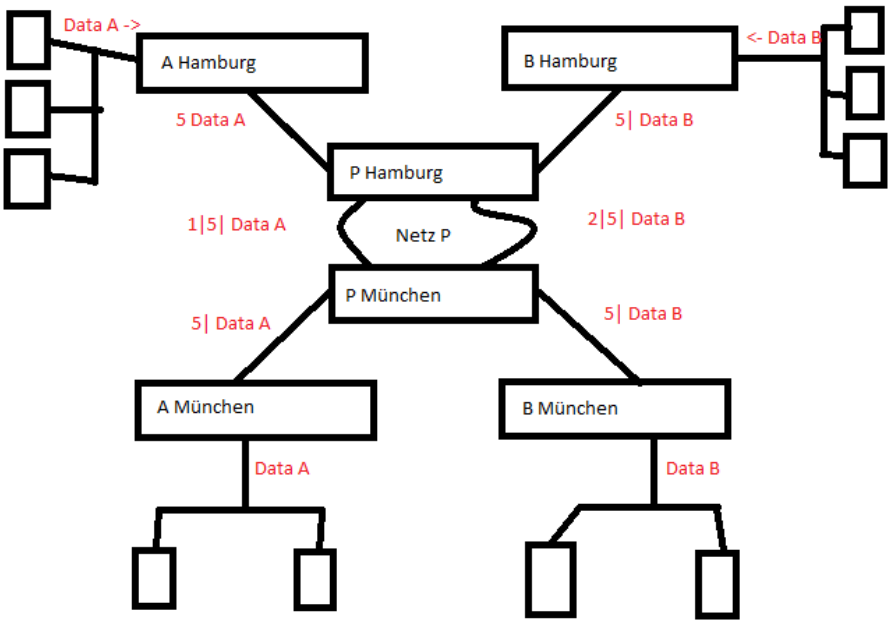


Abbildung 2.3: Netzaufbau

2.4.3 Carrier Ethernet

a) Abb.: 2.3

b) Abb.: 2.3

c) Abb.: 2.4

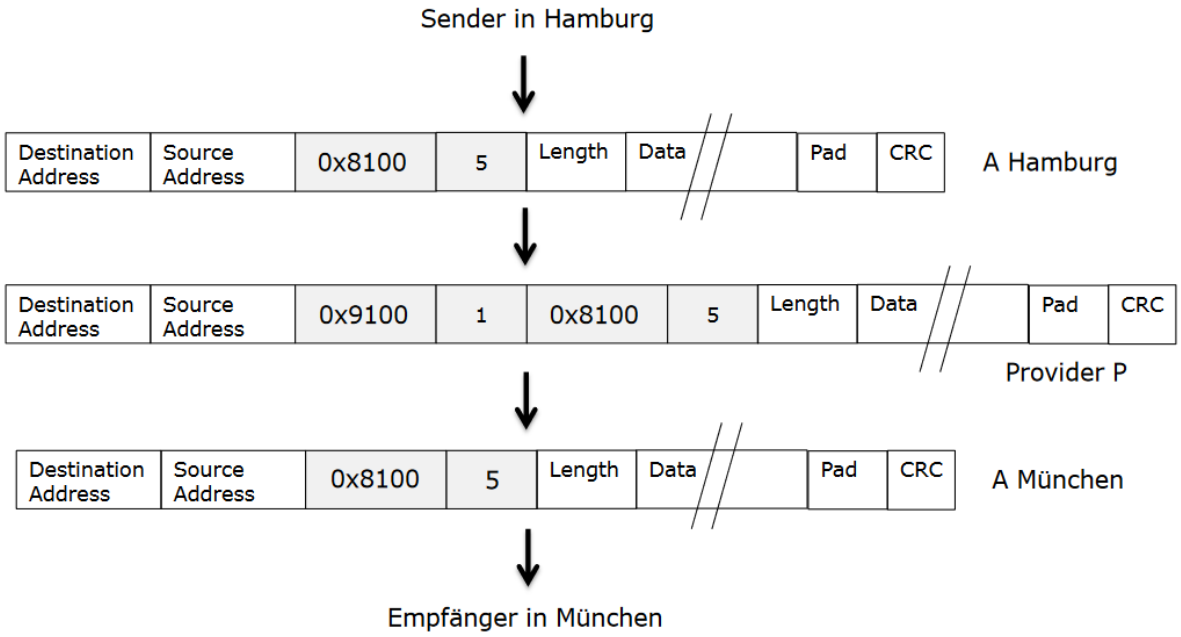


Abbildung 2.4: Ethernet Headerfeld

2.4.4 MPLS

zu Abb.: 2.5

	IN	DEST	OUT	
SF	(1,-) (1,-) (2,7) (3,10) (1,-) (2,14)	134.5.0.0/16 230.3.0.0/16 * * 134.5.42.0/24 *	(3,1) (2,2) (1,-) (1,-) (2,11) (1,-)	a) a) c) c) d) d)
H	(1,1) (3,9)		(3,4) (1,10)	a) c)
W	(1,4) (3,8)		(3,5) (1,)	a) c)
C	(1,2) (2,6) (1,11) (2,13)		(2,3) (1,7) (2,12) (1,14)	a) c) d) d)
NY	(2,5) (1,3) (4,-) (3,-) (1,12) (3,-)	* * 217.8.0.0/16 217.8.0.0/16 * 217.8.42.0/24	(3,-) (4,-) (1,6) (2,8) (3,-) (1,13)	a) a) c) c) d) d)

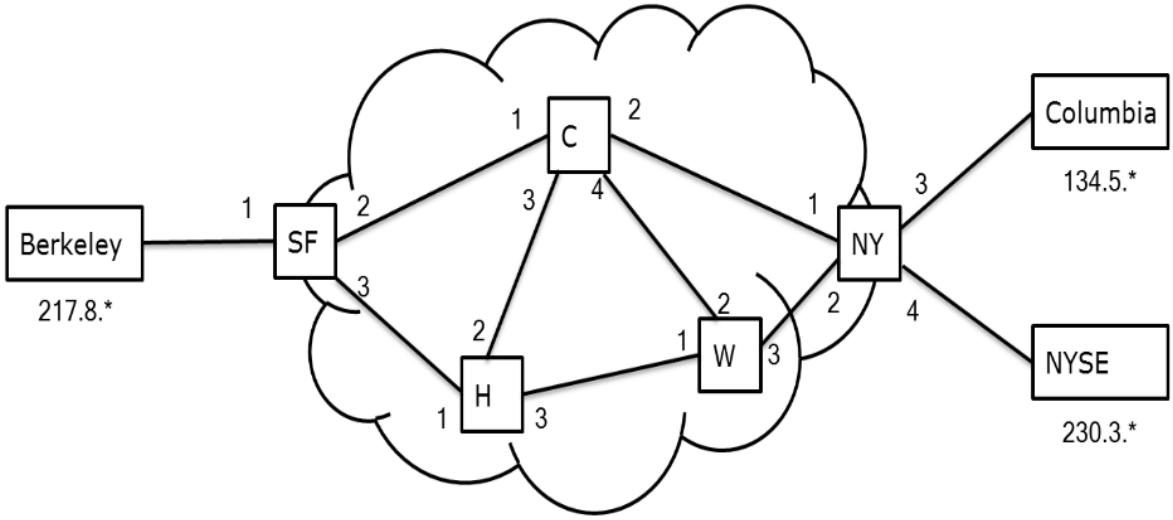


Abbildung 2.5: MPLS-Network

b)

Teilstrecke	Label	Ip-Dest
Berkeley - SF	-	134.5.20.217
SF - H	1	134.5.20.217
H - W	4	134.5.20.217
W - NY	5	134.5.20.217
NY - Columbia	-	134.5.20.217

2.4.5 SONET/SDH und OTN

Abb.:2.6

2.5 Sicherungsschicht

2.5.1 Rahmenbildung

a) RB = 01111110
RB|n Bits|RB

- T = 101 0100
- o = 110 1111

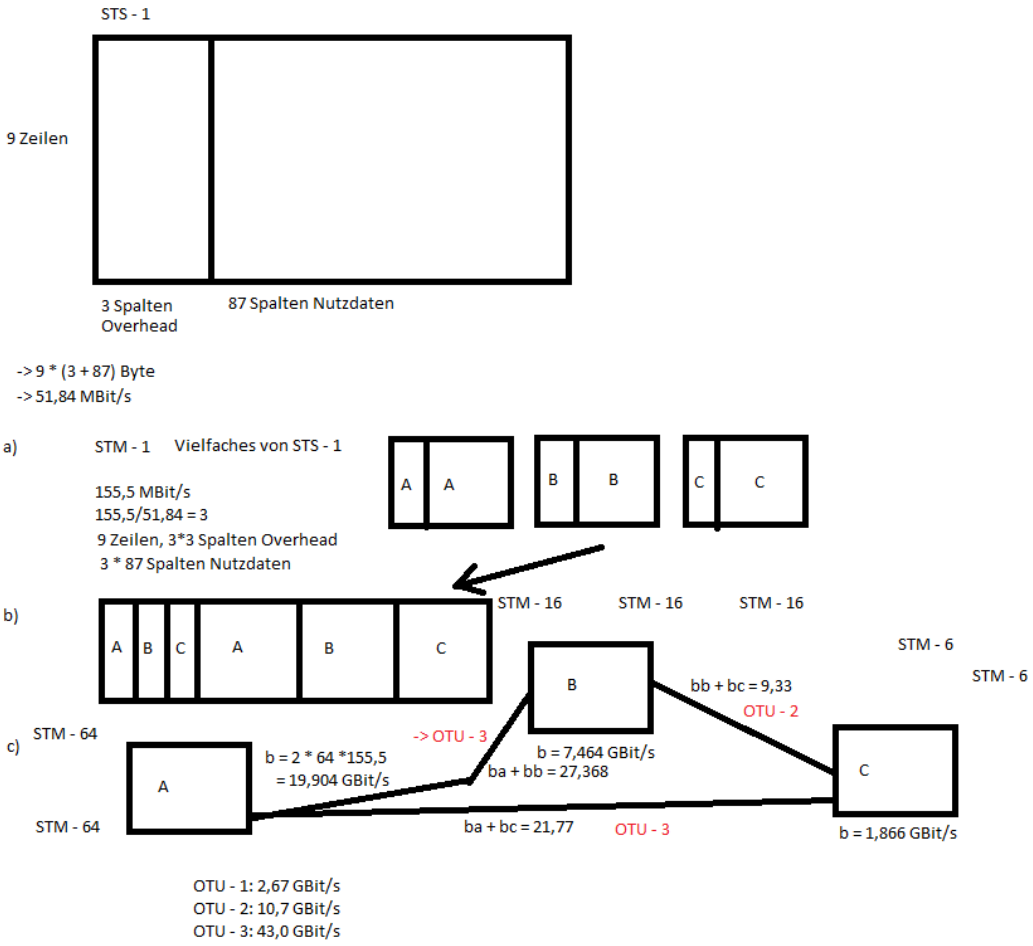


Abbildung 2.6: STS - 1 und STM - N ($n \in \mathbb{N}$)

- k = 110 1011
- e = 110 0101
- n = 110 1110

→ 01111110 101 0100 110 1111 110 1011 ...01111110

b) Rahmenbegrenzer teil der n Bits zwischen o und k

c) Bit-Stuffing: Jede Folge aus fünf 1en wird mit einer 0 erweitert, Empfänger löscht nach 111110 die 0.
Byte-Stuffing: ESC-Sequenz: A|RB|B → A|ESC|RB|B
A|ESC|B → A|ESC|ESC|B

2.5.2 Fehlerkorrigierende Codes

a) (1) B
(2) C
(3) B
(4) A oder B
(5) A

b) min(d(A,B),d(A,C),d(B,C)) = 3

c) $f_e = d - 1 = 2$
$f_k = \frac{d-1}{2} = 1$

Paritätsbit

a) (Un)gerade Parität: Paritätsbit wird so gewählt, dass die Anzahl an 1en (un)gerade ist.

(1) $|101100111|_1 = 6 \rightarrow$ Störung oder Par.Bit falsch berechnet
(2) $|101011100|_1 = 5 \rightarrow$ richtig übertragen oder gerade Anzahl Bitfehler

2.5.3 CRC

a) G(x): Generatorpolynom
r = grad(G(x))
P_D: Datenpolynom
m = grad(P_D)

$x^5 + x^4 + x^2 + 1 \rightarrow 110101$
r = 5

$$
\begin{array}{l}
101000110100000 \div 110101 \\
110101 \\
011101 \\
\quad 110101 \\
\quad 00111010 \\
\qquad 110101 \\
\qquad\quad 111110 \\
\qquad\quad 110101 \\
\qquad\quad 00101100 \\
\qquad\quad 00110101 \\
\qquad\quad 000110010 \\
\qquad\quad 000110101 \\
\qquad\quad 0000001110 = Rest \rightarrow 101000110101110
\end{array}
$$

b) $101000110101110 \div 110101 \rightarrow Rest = 0$

c)

$$
\begin{aligned}
b &= 40kBit/s \\
\tau &= 20ms \\
T &= t_s + 2\tau \\
&50\% Effizienz \\
t_s &= 0,5T \\
&= 0,5(t_s + 2\tau) \\
&= 0,5t_s + \tau \\
0,5t_s &= \tau \\
\Rightarrow t_s &= 2\tau \\
\frac{F}{b} &= 2\tau \\
F &= 2\tau b = 2 \cdot 20 \cdot 40 = 1600Bit
\end{aligned}
$$

2.6 Vermittlungsschicht

2.6.1 Shortest Path Algorithmus (Dijksstra-Algorithmus)

Wähle Startknoten als permanenter und Arbeitsknoten → Wähle als nächsten Arbeitsknoten den Knoten mit dem Geringsten Pfad und setzte ihn permanent → ...

a) A $\rightarrow$ F in Schritten

- permanenter Knoten A; Arbeitsknoten A
  - B: (A,3), C:(A,4)
- permanenter Knoten A,B; Arbeitsknoten B
  - D: (B,8), E:(B,5)
- permanenter Knoten A,B,C; Arbeitsknoten C
  - D: (C,5)
- permanenter Knoten A,B,C,E; Arbeitsknoten E
  - F: (B,11)
- permanenter Knoten A,B,C,E,D; Arbeitsknoten D
  - F: (D,9)

Der kürzeste Weg ist demzufolge: $A \rightarrow C \rightarrow D \rightarrow F$ mit Kosten 9. Tabellarisch mit Spalten Knotenvorrat(momentan erreichbare Knoten, 1.Sp: {B,C}), Arbeitsknoten (1.Sp: A), Wege (1.Sp: (AB,3) (AC,4)

b) Es gibt nur einen Pfad $\rightarrow$ ABEF(11) muss der kürzeste Weg sein

2.6.2 Einsatz von IP: Adressen und Subnetze

a) Aufteilung der IP-Adressen in Netzanteil und Geräteanteil $\Rightarrow$ Hierarchisches Routing:

- Subnetzadressen müssen außerhalb einer Organisation nicht bekannt sein
- Routingtabellen werden deutlich kleiner

b)

- Netzanteil = IP-Adresse AND Maske
- Geräteanteil = IP-Adresse AND NOT Maske
- 129.44.0.0/**16** $\rightarrow$ Maske: 1111111111111111 0...16x...0

$$
\begin{aligned}
A_1 &= 129.44.0.7 \\
M_1 &= 2255.255.128.0 \\
A_1 &= 10000001.00101100.00000000.00000111 \\
M_1 &= 11111111111111111000000000000000 \\
\rightarrow S_1 &= 10000001001011000000000000000000 \\
S_1 &= 129.44.0.0 \\
X &= 2^{15} - 2 = 2^{|\text{Geräteanteil}|} - 2 = 2^{32-|\text{Netzanteil}|} - 2
\end{aligned}
$$

reservierte Adressen: Geräteanteil = 0 ... 0 (Netzwerk) bzw. 1 ... 1 (Broadcast)

$$A_2 = 1000001.00101100.11100000.00001111$$
$$M_2 = 11111111111111111100000000000000$$
$$Y = 2^{14} - 2$$

2.6.3

- C ist Routng-Firewall
- INF1 und ZIH1 sind Default-Router, INF2 und ZIH2 sind Standby-Router

a) A → Router SyA → Firewall SyA → INF1 → ZIH1 → Internet → D → Internet → ZIH1 → INF1 → |A| → B

b)
- SyA: 141.76.40.0/22
- IAI: 141.76.82.0/24
- Firewalls: 141.76.29.32/28

Routing Tabellen:

- INF1
  - Typ
    - C
    - C
    - S
    - D
  - Filter
    - 141.76.29.32/28
    - 141.76.82.0/24
    - 141.76.40.0/22
    - 0.0.0.0/0
  - Ziel
    - Firewalls
    - IAI
    - 141.76.29.33
    - ZIH1
- Routing Firewall C
  - Typ
    - C

        * D
    - Filter
        * 141.76.40.0/22
        * 0.0.0.0/0
    - Ziel
        * SyA
        * INF1

c) dynamische Einträge (z.B. OSPF)

d) primär nach den Netzmasken (größte zuerst), sekundär nach IP-Adressen (kleinste zuerst) ⇒ Longest Prefix Match

e) dynamische Routen werden auf INF2 umgeleitet

2.6.4

a) Internet Control Message Protocol (ICMP), Type 8 (Echo(ping)request)

b)
- src: 192.168.1.102
- dst: 128.59.23.100

c) Ja. TimeToLive = 1 (TTL). Jeder Router verringert TTL um 1. Wenn TTL = 0, sendet Router eine ICMP Typ 11 (TTL exceeded)-Nachricht.

d) Fragmentierung

- erstes Fragment: more fragments(1) fragement offset(0)
- weiteres Fragment: more fragments(1) fragment offset (>0)
- letztes Fragment: morefragments(0) fragment offset (>0)
- hier: 0/0

⇝ nicht fragmentiert

e)
- Src und Dst bleiben gleich
- total length, TTL, headercheckcsum **können** sich ändern
- Identification **muss** sich ändern

2.6.5 IPv6

a)
- IPv4: 32-Bit-Adressen. Theoretisch $2^{32} = 4,3$ Mrd mögliche Adressen $\rightarrow$ nur ein Teil der Adressen wegen Subnetzeinteilung und reservierten Adressen nutzbar
- IPv6: 128-Bit-Adressen. Theoretisch ca $3,4 \cdot 10^{38}$ Adressen möglich

b)
- Vergrößerung des Adressraums (siehe a))
- Vereinfachung Header
- Vereinfachung mobiler Netzwerkteilnahme (Mobile IP siehe Ü12)
- Implementierung von Sicherheitsmerkmalen (IPsec)
- Qualitätsmerkmale werden unterstützt (siehe Ü10)

c) Berechnungen in Netzwerkgeräten i.d.R. aus Effizienzgründen in der Hardware realisiert $\rightarrow$ Umstellung von 32 auf 128 Bit ist von der Hardware abhängig

2.7 Transportschicht

fill siehe Tobi telegramm

2.8 Netzwerkperformance

2.8.1 Ehernet-Performance

a) (1) Fast Ethernet: Fmin = 64Byte, Fmax = 1518 Byte, SL = 8Byte, IFG = 12Byte
Berechnung η siehe Lösungen

(2) Gigabit Ethernet: Fmin = 512Byte, Fmax = 1518 , SL = IFG = 8

- Padding: 8(SL) 64(F) ...(PAD) 8(IFG)
Berechnung η(Pad + F auf 512) siehe Lösungen
- Bursting: 8(SL) 64(F) 8(IFG) ...(PAD) $\rightarrow$ ...
Berechnung η siehe Lösungen, n = 13 (aus Vorlesung)

- Jumbo-Frames: $F_{max} = 8192\text{Byte}$

$$\begin{aligned} n \cdot (SL + F + IFG9) + PAD &\leq 8192\text{Byte} \\ 80n &\leq 8192 - 448 = 7744 \\ n &\leq 96,8 \\ \rightsquigarrow n &= 96 \\ \eta_{\text{Frame}} &= \frac{n \cdot F}{n(SL + F + IFG) + PAD} \\ &= \frac{96 \cdot 64Byte}{96(8 + 64 + 8)\text{Byte} + 448\text{Byte}} \\ &= 75,59\% \end{aligned}$$

b)
- F = 128 Byte
- PAD = (512-128)Byte = 384Byte
- Fast Ehernet: η_{Frame} = 86,49%
- GB Eth. + PAD: η_{Frame} = 24,24%
- GB Eth. + PAD + BURST: η_{Frame} = 64,37%, n = 7
- Jumbo Frames: η_{Frame} = 84,71%, n = 54

2.8.2 Fairness

a) Fairness bezeichnet den gleichberechtigten und gleichmäßigen Zugriff aller Teilnehmer eines Netzwerkes auf die vorhandenen Ressourcen.

- unfair:

b) fair: Server vergibt Wartenummern. DriveRequest 0: Wartenummer zieehen[Wait(WN)] DriveRequest WN : Go wenn WN richtig

2.8.3 Überlastungsüberwachung

a)
- Nachfrage: x1 = 6, x2 = 6,5, x3 = 12, x4 = 15, K = 30 $\rightarrow \frac{30}{4}$
- Runde 1: k1 = 6, k2 = 6,5, k3 = 7,5, k4 = 7,5, Ü1 = 1,5, Ü2 = 1 K = 2,5$\rightarrow \frac{2,5}{2}$
- Runde 2: k3 = 8,75, k4 = 8,75

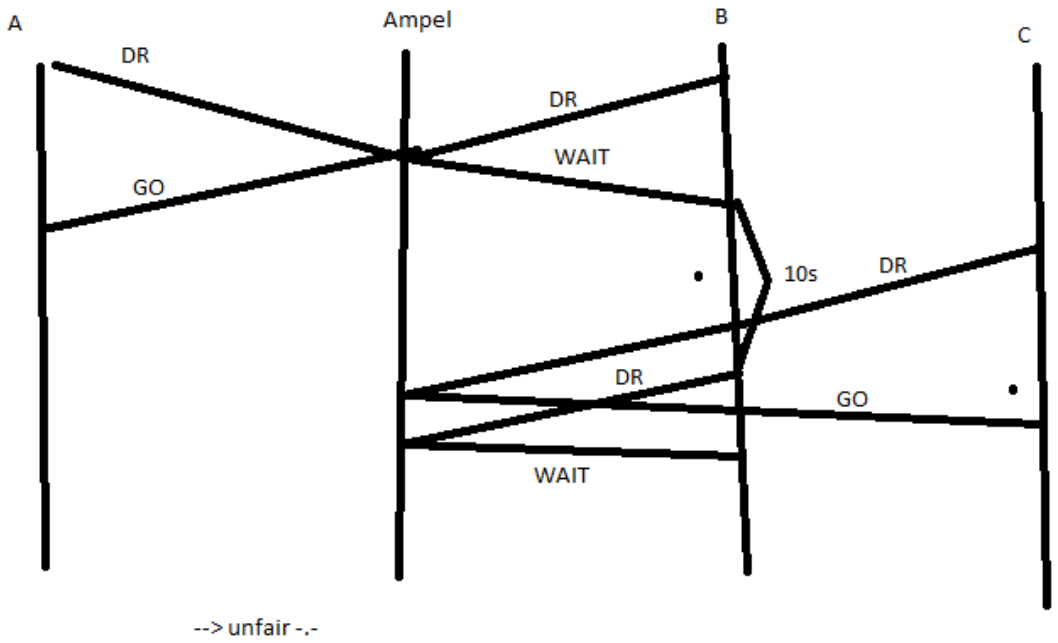


Abbildung 2.7: unfair

b) $K = 16$, $g_{\text{ges, ungesättigt}} = \Sigma g_i k_i^* = K \cdot \frac{g_i}{g_{\text{ges, ungesättigt}}}$

$$g_{\text{ges, ungesättigt}} = 2,5 + 4 + 0,5 + 1 = 8$$

$$k_1^* = 16 \cdot \frac{2,5}{8} = 5$$

$$k_2^* = 16 \cdot \frac{4}{8} = 8$$

$$k_3^* = 16 \cdot \frac{0,5}{8} = 1$$

$$k_4^* = 16 \cdot \frac{1}{8} = 2$$

RUNDE 1:$x_1 = 4, x_2 = 2, x_3 = 10, x_4 = 5$

$k_1 = 4, k_2 = 2, k_3 = 1, k_4 = 2, U_1 = 1, U_2 = 6$

$K = U_1 + U_2 = 7$

$g_{\text{ges, ungesättigt}} = 0,5 + 1 = 1,5$

$$k_3^* = 7 \cdot \frac{0,5}{1,5} = 2,33$$

$$k_4^* = 7 \cdot \frac{1}{1,5} = 4,67$$

RUNDE 2:$k_3 = 3,33 k_4 = 5, U_4 = 1,67$

RUNDE 3:$k_3 = 5 \rightarrow k_3$ ist ungesättigt

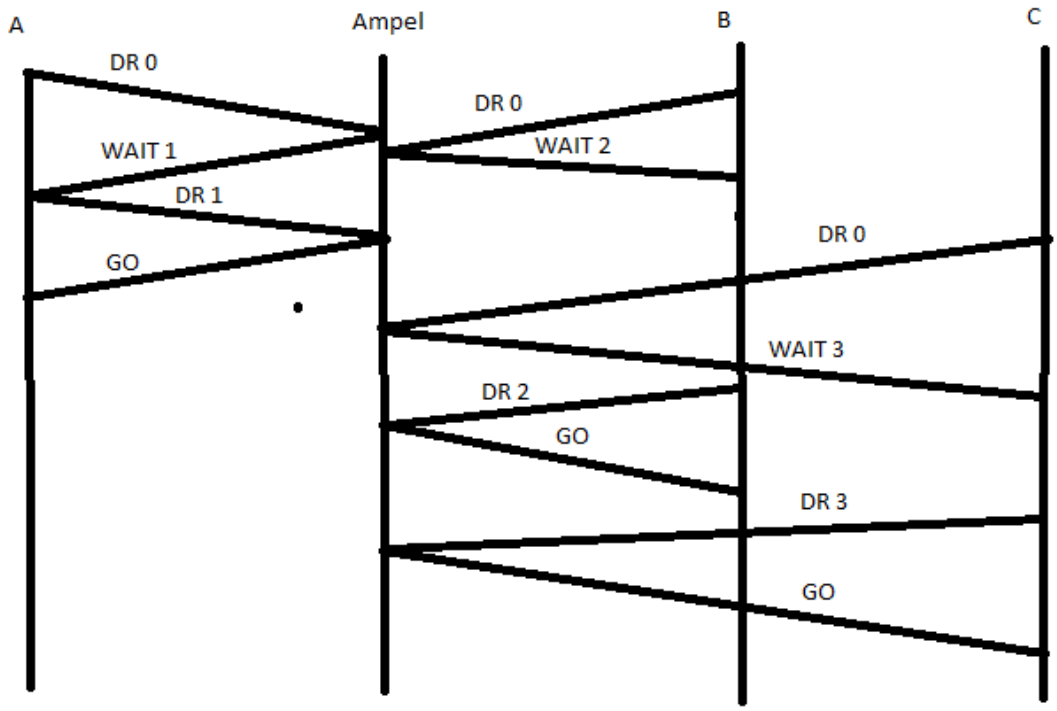


Abbildung 2.8: fair

2.8.4 Überlaststeuerung

$\text{Last}_{\text{neu}} = a \cdot \text{Last}_{\text{alt}} + (1 - a) \cdot \text{Last}_{\text{aktuell}}$
$a \ldots$ Anpassungsfaktor, $a = 0,3$ $\text{Last}_{\text{alt}} = 0$
$\text{Last}_{\text{aktuell}}$
$1 : \text{Last}_{\text{neu}} = 0,3 \cdot 0 + (1 - 0,3) \cdot 1 = 0,7$
$5 : \text{Last}_{\text{neu}} = 0,3 \cdot 0,7 + 0,7 \cdot 5 = 3,71$
$8 : \text{Last}_{\text{neu}} = 0,3 \cdot 3,71 + 0,7 \cdot 8 = 6,71$
$9 : \text{Last}_{\text{neu}} = 0,3 \cdot 6,71 + 0,7 \cdot 9 = 8,31 \rightarrow$ **CHOKE**
$9 : \text{Last}_{\text{neu}} = 0,3 \cdot 8,31 + 0,7 \cdot 9 = 8,79 \rightarrow$ **CHOKE**
$7 : \text{Last}_{\text{neu}} = 0,3 \cdot 8,79 + 0,7 \cdot 7 = 7,54$
$2 : \text{Last}_{\text{neu}} = 0,3 \cdot 7,54 + 0,7 \cdot 2 = 3,66$

2.8.5 Leistungssteigerung

a) BDP: Anzahl an Daten (in Bit), die auf einer Leitung liegen:
$BDP = b \cdot \tau$

- wird weniger gesendet, als das BDP, ist die Leitung nicht ständig belegt → Empfängerfenster ≤ BDP
- Go-Back-N: bleibt eine Bestätigung aus, muss der gesamte Leitungsinhalt wiederholt werden

b) $v = \frac{d}{\tau} \rightsquigarrow \tau = \frac{d}{v} = \frac{4000 \text{ km}}{200000 \frac{\text{km}}{\text{s}}}$
$BDP = b \cdot \tau = \frac{4000 \text{ km}}{200.000 \frac{\text{km}}{\text{s}}} \cdot 1 \frac{\text{GBit}}{\text{s}} = 20 \text{ MBit} = 2,5\text{MB}$
keine Bedeutung für die Framegröße

c) $BDP = 2 \cdot b \cdot \tau = 2 \cdot 640 \frac{GBit}{s} \frac{8000 \text{ km}}{200000 \frac{\text{km}}{\text{s}}}$
$= 51,2GBit = 6,4GB > 650MB$

2.9 Internetdienste

2.9.1 Domain Name System (DNS)

a) $\underbrace{\text{www.}}_{hostname}\ \underbrace{\text{inf.}}_{subdomain}\ \underbrace{\text{tu-dresden.}}_{domain}\ \underbrace{\text{de}}_{topleveldomain}$

b) (1) DNS Client ↔ DNS Server
- Client sendet den Namen, zu dem IP benötigt wird, an seinen Default-DNS-Server
- Server kümmert sich um komplette Adressermittlung
- ermittelte Adresse wird an Client übermittelt

(2) DNS Server ↔ DNS Server
- falls IP für DNS-Server unbekannt: Weiterleitung der Anfrage an hierarchisch höheren/niedrigeren Server

c)

NAME	TTL(Sek.)	CLASS	TYPE	VALUE
jupiter (relative Adressierung)	86.400	IN	A	117.186.1.1
jupiter	86.400	IN	AAAA	2001:db8: 85a3:8d3::1
saturn	86.400	IN	A	117.186.1.2
saturn	86.400	IN	AAAA	. . . ::2
rn-edu.de.	86.400	IN	NS	sonne
sonne	86.400	IN	A	117.186.1.3
sonne	86.400	IN	A	. . . ::3
_ sip._ tcp.rn-edu.de.	86.400	IN	SRV	0 0 5060 mond. rn-edu.de.
_ sip._ udp.rn-edu.de.	86.400	IN	SRV	0 0 5060 mond. rn-edu.de.
mond	86.400	IN	A	117.186.1.4
mond	86.400	IN	AAAA	2001: . . . ::4

- A = IPv4 - Adresse
- AAAA = IPv6-Adresse
- SRV = Service
- NS = Name Server

2.9.2 E-Mail

Base64: Binary-to-Text-Encoding
Vorgehensweise: je 3 Byte werden in 4 6-Bit-Blöcke aufgeteilt. Jeder 6-Bit-Block wird mit einem ASCII-Zeichen codiert (8 Bit). Ggf. werden Füllbytes angehängt.
Ausgangsgröße von 20.000 Byte (ab hier alles in Byte)
Anzahl 3er-Gruppen: $\frac{20000}{3} = 6666,67 \rightarrow 6667$ Gruppen
Anzahl Bytes (Base64): $6667 \cdot 4 = 26668$
Anzahl Zeilenumbrüche: $\lceil \frac{26668}{76} \rceil = 351$
Gesamtzahl Bytes: $26668 + 351 \cdot 2 = 27370$

2.9.3 World Wide Web

a) (1) Position im HTML-Dokument feststellen

(2) Auswertung der Anweisung <a href = "http://www. . . . » link

(3) Zielserver muss aus URL ermittelt werden
→ DNS-Anfrage (Dienst)

(4) www-Seite mittels HTTP anfordern (Protokoll)

(5) HTTP-Response auswerten, Seite darstellen

b) (1) gaia.cs.umass.edu/cs451/index.html
(2) HTTP/1.1
(3) Ja. → Connection: Keep-Alive
(4) unbekannt

c) (1) Ja. → 200 OK
(2) 12:39:45GMT
(3) 18:27:46GMT
(4) 3874
(5) <!doc ...
(6) Ja. → Connection: Keep-Alive

d) (1) Dienste:
- OpenURL(String URL)
- ReceivedRessource (buffer res)

(2) Protokoll:
- Host-Teil aus URL ermitteln
- IP-Adresse über DNS ermitteln
- GET Request an Server senden
- bei Empfang von "200 OK": ReceivedRessource() aufrufen

2.9.4 Management in Rechnernetzen - Simple Network Management Protocol (SNMPv3)

SNMP-Nachrichten-Typen

- GET
- GETNEXT
- GETBULK
- SET
- RESPONSE
- TRAP

a)

Manager		Agent
GetReq -> <- Get CNF	GET -> <- RESPONSE	GetInd -> <- GetRes

b)

Manager		Agent
	GET -> <- RESPONSE GET -> <- RESPONSE GETNEXT -> <- RESPONSE	

c)

Manager		Agent
TrapInd	<- TRAP	TrapReq

d)

Manager		Agent
SetReq <- SetCnf	SET -> <- RESPONSE	SetInd -> <- SetResp

2.9.5 Zusammenspiel von UDP, TCP, HTTP und DNS

	Absender	Ziel	Protokoll	Meth/Antw	Inhalt
1	C	D	DNS/UDP	A-Request	URI von W
2	D	C	DNS/UDP	A-Response	IP von W
3	C	W	TCP	SYN	Verb.Aufbau
4	W	C	TCP	SYN-ACK	Verb.Aufbau
5	C	W	TCP	ACK	Verb.Aufbau
6	C	W	HTTP/TCP	GET	Ziel-Uri
7	W	C	HTTP/TCP	200 OK	Website
8	C	W	TCP	FIN	Verb.Abbau
9	W	C	TCP	FIN-ACK	Verb.Abbau
10	C	W	TCP	ACK	Verb.Abbau

Teil II

Theoretical Informatic and Logic

Kapitel 3

Vorlesung? S. 22

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 $p(t_1, \ldots, t_n)$

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

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:

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

- **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

3.7 Äquivalenz und Normalenform (2)

3.7.1 Klauselform

- Sei F ein Satz der Prädikatenlogik und H eine Skolem-Normalform von F
- F ist erfüllbar gdw F erfüllbar
- H ist von der Form $\forall G = (\forall X_1) \dots (\forall X_n) G$, wobei $X_1, \dots, X_n$ alle in H vorkommenden Variablen sind
- In der Matrix G kommen keine Quantoren mehr vor
- Alle in G vorkommenden Variablen sind universell quantifiziert
- Wir können G in Klauselform transformieren
- Sei G' eine Formel in Klauselform, die semantisch äquivalent zu G ist
  - F ist erfüllbar gdw $\forall G'$ ist erfüllbar
  - F ist unerfüllbar gdw $\forall G'$ ist unerfüllbar

3.7.2 maschinelles Beweisen mathematischer Sätze

siehe Folien

3.7.3 Unifikation

Definition 4.40

Eine **Gleichung** ist ein Ausdruck der Form $s = t$ wobei s und t Terme sind

Definition 4.41

Ein **Unifikationsproblem** U besteht aus einer Mulitmenge von Gleichungen

$$\{s_1 = t_1, \ldots, s_n = t_n\}$$

und ist die Frage, ob es eine Substitution σ gibt, so dass $s_i\sigma = t_i\sigma$ für alle $1 \leq i \leq n$ gilt. Wenn es eine solche Substitution σ gibt, dann

- heißt U lösbar
- sind die Terme s_i und $t_i, 1 \leq i \leq n$, simultan unifizierbar
- ist σ ein Unifikator für U

3.7.4 Allgemeinste Unifikatoren

Definition 4.42

Seien σ und θ Substitutionen. σ ist **allgemeiner als** θ, symbolisch $\sigma \geq \theta$, wenn es eine Substitution λ gibt, s dass $\sigma\lambda = \theta$ gilt

Definition 4.43

Substitutionen σ und θ sind **Varianten**, symbolisch $\sigma \sim \theta$, wenn $\sigma \geq \theta$ und $\theta \geq \sigma$ gilt

Definition 4.44

Sei U ein Unifikationsproblem . Eine Substitution σ ist ein **allgemeinster Unifikator** für U, wenn σ ein Unifikator für U ist und $\sigma \geq \theta$ für jeden Unifikator θ für U gilt.
Im Englischen werden allgemeinste Unifikatoren Most General Unifier genannt und mit mgu abgekürzt.

Der Unifikationssatz

Satz 4.45

Wenn U ein lösbares Unifikationsproblem ist, dann gibt es einen allgemeinsten Unifikator für U

Beweis Satz 4.45

- Unifikationsalgorithmus spezifizieren
- Terminierung beweisen
- Korrektheit beweisen

3.8 Äquivalenz und Normalform

3.8.1 Beweis Satz 4.45

Der Unifikationsalgorithmus

Eingabe

Ein Unifikationsproblem U

Ausgabe

Ein allgemeinster Unifikator θ für U, wenn U lösbar ist, oder **nicht unifizierbar**, wenn U keine Lösung besitzt

Terminierung (76)

Korrektheit (79)

3.8.2 Allgemeinste Unifikatoren

Korollar 4.46

Sei U ein Unifikationsproblem und σ sowie θ allgemeinste Unifikatoren für U. Dann gilt $\sigma \sim \theta \rightarrow$ Allgemeinste Unifikatoren sind also bis auf Variantenbildung eindeutig $\rightarrow$ Man spricht deshalb häufig auch von **dem** mgu zweier Terme

3.9 Resolution

3.9.1 Resolutionsregel

Definition 4.47

Gegeben seien die prädikatenlogischen Klauseln

$$fill$$

mit $k, m, n \geq 0$. Wenn $\{s_i = t_i | 1 \leq i \leq k\}$ mit einem allgemeinsten Unifikator σ unifizierbar ist, dann nennen wir

$$C = \{L_1, \ldots, L_n\}\sigma\}$$

eine Resolvente C_1 und C_2 bezüglich $p(s_1, \ldots, s_k)$ und $\neg p(t_1, \ldots, t_k)$
fill

Definition 4.48 Faktorisierungsregel

Gegeben sei die prädikatenlogische Klausel:

$$C = fill$$

beziehungsweise

$$C = fill$$

mit $k, m \geq 0$ ... fill

3.9.2 Resolutionsableitungen und -widerlegungen

Definition 4.49

Sei $F = \forall < C_1, \ldots, C_n >$ eine prädikatenlogische Formel in Klauselform, wobei $C_i, 1 \leq i \leq n$, Klauseln sind.

$$fill$$

Beachte

- Eine Resolvente kann aus zwei Varianten einer Klausel gebildet werden.
- fill
- fill
- fill

Eine Resolvente kann aus zwei Varianten einer Klausel gebildet werden.

Vollständigkeit des Resolutionsverfahrens - Überblick

$\{F_1, \ldots, F_n\} \vDash F$ Satz 3.17 erweitert

$fill$

3.9.3 Herbrand-Interpretationen

Seien$F = \{s/0, f/1\}, R = \{p/1, q/1, r/1\}, X, Y \in V$

$$G = (\forall X)(\forall Y) < [p(X), q(X)], [r(f(Y))] >$$

$fill$

Definition 4.58

Sei $I = (D, \cdot^I)$ eine Interpretation. Eine zu I ... fill

Lemma 4.59

Wenn eine Interpretation I Modell für einen prädikatenlogischen Satz F in Skolem-Normalform ist, dann ist auch jede zu I korrespondierende Herbrand-Interpretation ein Modell für F
Folgerung Die reellen Zahlen lassen sich in der Prädikatenlogik nicht charakterisieren

- Angenommen, die reellen Zahlen liesen sich charakterisieren
- Dann gibt es Satz f, so dass für alle $I = (D, \cdot^I)$ gilt: wenn $F^I = \top$ dann ist D überabzählbar
- Wähle I mit $F^I = \top$
- Sei J korrespondierende Herbrand-Interpretation
- fill

Satz 4.60

Eine prädkatenlogischer Satz F in Skolem-Normalform ist unerfüllbar gdw F wird von jeder Hebrand-Interpretation auf $\perp$ abgebildet
Beweis siehe Folien

3.10 Beweis

3.10.1 Beweis Satz 4.60

siehe Skript

3.11 Ende Logik Anfang Automaten

Skript Baader:
https://lat.inf.tu-dresden.de/teaching/ss2014/THEOLOG/
THEOLOG2014 B626165

3.12 02.06.2016

3.12.1

Beispiel 3.4 (Addition)

Die Addition natürlicher Zahlen kann durch primitive Rekursion wie folgt definiert werden.:

$$\begin{aligned} add(x,0) &= x && = g(x) \\ add(x,y+1) &= add(x,y)+1 && = h(x, add(x,y), y) \end{aligned}$$

Das heißt also: *add* entsteht durch primitive Rekursion aus den Funktionen

- $g : \mathbb{N} \to \mathbb{N}$
  fill

3.12.2 Definition

3.12.3 primitiv rekursive Funktion: Beispiel

Multiplikation erhält man durch primitive Rekursion aus der Addition:

$$\begin{aligned} mult(x,0) &= 0 && = null^{(1)}(x) \\ mult(x,y+1) &= add(x, mult(x,y)) && = add(\pi_1^{(3)}, \pi_2^{(3)})(x, mult(x,y), y) \end{aligned}$$

Exponentiation erhält man durch primitive Rekursion aus der Multiplikation

$$\begin{aligned} exp(x,0) &= 1 && = s(null^{(1)}(x)) \\ exp(x,y+1) &= mult(x, exp(x,y)) && = ... \end{aligned}$$

Die Funktion

$$min1 : \mathbb{N} \to \mathbb{N} \text{ mit } x \mapsto x-1 := \begin{cases} x-1 & x > 0 \\ 0 & x = 0 \end{cases}$$

ist ebenfalls primitiv rekursiv:

$$\begin{aligned} min1(0) &= 0 && = null^{(0)}() \\ min1(y+1) &= y && = ... \end{aligned}$$

3.12.4 Besipliel 3.7

Aus den bisher betrachteten Funktionen erhält man damit die Funktion

$$c : \mathbb{N}^2 \to \mathbb{N} \text{ mit } (x, y) \mapsto 2^x \cdot (2y + 1) - 1$$

durch Komposition, d.h. c ist primitiv rekursiv (diese Funktion ist eine Bijektion von $\mathbb{N}^2 \to \mathbb{N}$). Da c eine Bijektion ist, gibt es die Umkehrfunktionen c_0 und c_1 mit der Eigenschaft:

$$c_0(c(x, y)) = x \text{ und } c_1(c(x, y)) = y$$
$$c(c_0(z), c_1(z)) = z$$

Diese ergeben sich im Prinzip aus dem Beweis von Lemma 3.8:

$$c_0(z) = ...$$
$$...$$

3.12.5 Lemma 3.9

Die Funktionen sign: $\mathbb{N} \to \mathbb{N}$ und $\overline{sign} : \mathbb{N} \to \mathbb{N}$ mit:

$$sign(x) = \begin{cases} 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

3.12.6 Fallunterscheidung

Es seien $g_1, g_2, h : \mathbb{N}^n \to \mathbb{N}$ gegeben. Die Funktion $f : \mathbb{N}^n \to \mathbb{N}$ entsteht daraus durch Fallunterscheidung, falls für alle $x \in \mathbb{N}^n$ gilt:

$$f(x) = \begin{cases} g_1(x) & \text{falls } h(x) = 0 \\ g_2(x) & \text{falls } h(x) > 0 \end{cases}$$

3.12.7 Lemma 3.11

sind g_1, g_2, h primitiv rekursiv, so auch f.

3.12.8 Definition 3.12: bechränkte Minimalisierung

Es sei $n \geq 0$. Die Funktion $f : \mathbb{N}^{n+1} \to \mathbb{N}$ entsteht aus $g : \mathbb{N}^{n+1} \to \mathbb{N}$ durch beschränkte Minimalisierung, falls gilt:

$$f(x, y) = \begin{cases} j & \text{falls } j = min\{i \leq y | g(x, i) = 0\} \text{ existiert} \\ y + 1 & \text{sonst} \end{cases}$$

Wir schreiben dann $f = \overline{\mu} g$.

3.12.9 Lemma 3.13

Ist g primitiv rekursiv, so auch $\overline{\mu}g$.

3.13 •

Es sei P ein LOOP - Programm für $f : \mathbb{N}' \to \mathbb{N}$ Es sei l der maximale Index der in P vorkommenden Variablen und $K = max\{r, l\}$. wir zeigen durch Induktion über den Aufbau von LOOP-Programmen, dass die folgende Funktion $g_p : \mathbb{N} \to \mathbb{N}$ primitiv rekursiv ist:

$$z \to_{\text{kodieren}} a_0, \ldots, a_k \to_P b_0, \ldots, b_k \to_{\text{dekodieren}} g_{P^{(z)}}$$

... fill ...

Um alle berechnbaren Funktionen zu erhalten, muss man

- die primitiv rekursiven Funktionen um eine weitere Operation, die unbeschränkte Minimalisierung erwetiern;
- die LOOP-Programme muss an um eine While-Schleife erweitern ...

Beachte: Der μ-Operator sucht im Prinzip nach dem kleinsten y, so dass $g(\underline{x}, y) = 0$ ist. Dabei müssen aber alle vorherigen Werte g...

3.13.1 Definition 4.1

Die funktion $f : \mathbb{N}^n \to \mathbb{N}$ entsteht aus $g : \mathbb{N}^{n+1} \to \mathbb{N}$ durch unbeschränkte Minimalisierung ...

3.13.2 Beispiel 4.2

$$g_1(x, y) = \begin{cases} x - y & \text{falls } x \geq y \\ \text{undefiniert} & \text{falls } x < y \end{cases}$$

$$g_2(x, y) = \begin{cases} y - x & \text{falls } x \geq y \\ \text{undefiniert} & \text{falls } x < y \end{cases}$$

$$g_3(x, y) = x + y = \begin{cases} 0 & \text{falls } x = 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Durch Anwendung ...

3.13.3 Definition 4.3 Klasse der μ-rekursiven Funktionen

fill

3.13.4 Definition 4.4 Syntax von WHILE-Programmen

Die Syntax von WHILE-Programmen enthält alle Konstrukte in der Syntax von LOOP-Programmen und zusätzlich

4. Falls P ein WHILE-Programm ist und $i \geq 0$, so ist auch

$$\text{WHILE } x_i \neq 0$$

fill
Beacte: Man könnte bei der Definition der While-Programme auf das LOOP-Konstrukt verzichten, da es durch WHILE simulierbar ist:

```
LOOP x DO P END
```

kann simuliert werden durch:

```
y = x + 0
WHILE ...
```

fill

3.13.5 Definition 4.5

Die (partielle) Funktion $f : \mathbb{N}^2 \to \mathbb{N}$ mit:

$$(x, y) \mapsto \begin{cases} x - y & \text{falls } x \geq y \\ \text{undefiniert} & \text{sonst} \end{cases}$$

ist WHILE-berechenbar durch das folgende Programm:

```
WHILE x2 != 0 DO
        x3 = x1;
        WHILE x3 != 0 DO
                x1 = x1 - 1
                x2 = x2 - 1
                x3 = 0
        END
END
x0 = x1;
```

3.13.6 Satz 4.7

Die Klasse der μ ... fill ...

3.13.7 Beweis Satz 4.7

3.14

fill

3.14.1 Darstellung von Zahlen auf Touringmaschinen

fill
Ändern dieses Symbols zu a_j:
Der neue wert von x_3 ist

$$(j_r, \ldots, j_2, j)_b = div((j_r, \ldots, j_2, j_1)_b, b) \cdot b + j$$

Das WHILE-Programm, welches die gegebene DTM simuliert, arbeitet wie folgt:

- Aus der Eingabe wird die Kodierung der Startkonfiguration der DTM in den Variablen ... fill
- fill
- fill

Insgesamt haben wir also gezeigt:

3.14.2 Theorem 4.10

Die folgenden Klassen von Funktionen ... fill

3.14.3 Universelle Turing-maschinen und unentscheidbare Probleme

Wir werden heir zeigen, dass es Relationen(Problee) gibt, die nicht Turing-entscheidbar sind, d.h. ihre charakteristische Funktion ist nicht Turing-berechenbar. Mit der Churchschen These ... fill

Konventionen

- Arbeitsalphabete der betrachteten Turingmaschinen sind Teilmengen von $\{a_1, a_2, a_3, \ldots\}$. wobei $a_1 = a, a_2 = b, a_3 = \not b$
- Zustandsmengen der etrachteten Turingmaschinen sind Teilmengen von $\{q_1, q_2, \ldots\}$ ... fill

3.14.4 Definition 5.1 - Kodierung einer Turingmaschine

Es sei $A = (Q, \Sigma, \Gamma, q_1, \Delta, F)$ eine Turingmaschine die (o.B.d.A.) die obigen Konventionen erfüllt.

- Eine Transition

  $$t = (q_i, a_j, a_{j'}, l|r|n, q_{i'})$$

  wird kodiert durch:

  $$code(t) = a^i b a^j b a^{j'} b a | a a | a a a b a^{i'} b b$$

- fill

3.14.5 Bemerkung 5.2

Es sei A eine TM und $w \in \Sigma^*$. Für

$$x = code(A)w$$

gilt das folgende:

- w beginnt genau nach dem zweiten Block bbb
- w kann also aus x eindeutig wieder herausgelesen werden

Es gibt eine DTM A_{CODE}, welche ...fill

3.14.6 Satz 5.3 Turing

Es gibt eine unverselle DTM U über Σ, d.h. eine DTM it der folgenden Eigenschaft: Für alle DTM A und alle $w \in \Sigma^*$gilt:

U akzeptiert $code(A)w$ gdw ... fill

Konfigurationskodierung:

fill

Arbeitsweise von U

fill

3.14.7 Satz 5.4

Die Relation

$$UNIV := code(A)w \in \Sigma^* | A \text{ is DTM über } \Sigma \text{ ... fill ...}$$

ist partiell entscheidbar aber unentscheidbar.

Beweis

fill

3.15

3.15.1 Satz 5.5

$L_1 \subset L_0$

Beweis

siehe Folien S. 43

3.15.2 Satz 5.6

Das Wortproblem für DTM (und damit auch für L_0 ist unentscheidbar, d.h. es gibt kein Berechnungsverfahren, das zu jeder gegebenen DTM A und jedem Eingabewort w entscheidet, ob A das Wort w akzeptiert.

3.15.3 Beweis

Wenn das Wortproblem für DTM entscheibar wäre, so wäre auch UNIV entscheidbar (ist es aber nicht). Um zu entsscheiden ob $code(A)w \in$ UNIV ist, müsste man ja nur das Entscheidungsverfahren für das Wortproblem feststellen lassen, ob A das Wort w akzeptiert.

3.15.4 Satz 5.7

Das Halteproblem für DTM ist unentscheidbar, d.h. es gibt kein Berechnungsverfahren das zu jeder gegebenen DTM $\hat{A}$ entscheidet, ob $\hat{A}$ beginnend mit leerem Eingabeband terminiert.

Beweis

Wäre das Halteproblem entscheidbar, so auch das Wortproblem.
Um zu gegebner TM A und gegebenem Wort w zu entscheiden, ob A das Wort w akzeptiert, könnte man dann nämlich wie folgt vorgehen:

- Bei leerem Band schreibt $\hat{A}$ zunächst w auf das Band.
- Danach verhält sich $\hat{A}$ wie A
- Stoppt A mit akzeptierender Stoppkonfiguration, so stoppt auch $\hat{A}$. Hält A mit nicht-akzeptierender Stoppkonfiguration, so geht $\hat{A}$ in eine Endlosschleife.

Damit gilt:

$$\hat{A} \text{ hält mit leerem Eingabeband} \qquad \underline{\text{gdw.}}\ A \text{ akzeptiert } w$$

Mit dem Entscheidungsverfahren für das Halteproblem (angewandt auf $\hat{A}$) könnte man also das Wortproblem (Ist w in $L(A)$?) entscheiden.

3.15.5 Satz 5.8

Das Leerheitsproblem für DTM (und damit auch für L_0) ist unentscheidbar, d.h. es gibt kein Berechnungsverfahren, das bei gegebener DTM $\hat{A}$ entscheidet, ob es eine Eingabe w gibt, auf der $\hat{A}$ terminiert.

Beweis

Wäre das Leerheitsproblem entscheidbar, so auch das Halteproblem.
Um bei gegebener DTM A zu entscheiden, ob A auf leerer Engabe hält, konstruiert man die DTM $\hat{A}$ wie folt:

- $\hat{A}$ löscht seine Eingabe
- Danach verhält sich $\hat{A}$ wie A
- Stoppt die Berechnung, so geht $\hat{A}$ in eine akzeptierende Stoppkonfiguration

Offenbar gibt es eine Eingabe, für die $\hat{A}$ hält <u>gdw.</u> A auf leerem Eingabeband hält.

3.15.6 Satz 5.9

Das Äquivalenzproblem für DTM (und damit auch für L_0) ist unentscheidbar.

Beweis

Offenbar kann man leicht eine DTM $\hat{A}$ konstruieren mit $L(\hat{A}) = \emptyset$.
Wäre das Äquivalenzproblem

$$L(A_1) = L(A_2)$$

entscheidbar, so könnte man durch den Test

$$L(A) = L(\hat{A})$$

das Leerheitsproblem für A entscheiden.

3.15.7 Satz 5.10

L_0 ist nicht unter Komplement abgeschlossen.

Beweis

Wir wissen von der in Satz 5.4 eingeführten Sprache UNIV:

- UNIV ist partiell entscheidbar, d.h. gehört zu L_0
- UNIV ist nicht entscheidbar

Wäre $\overline{UNIV} \in L_0$, d.h. partiell entscheidbar, so würde aber mit Satz 2.4 (Teil 4) folgen, dass UNIV entscheidbar ist.

Unentscheidbarkeit zeigen

Wie kann man Unentscheidbakeit eines Problems (formal: einer Relation) zeigen?

1 Durch ein Diagonalisierungsargument wie im Beweis von Satz 5.4

2 Das in den Beweisen der Sätze 5.6 bis 5.9 gewählte Vorgehen nennt man Reduktion:

   - Ein Problem P_1 (z.B. halteproblem) wird auf ein Problem P_2 (z.B. Äquivalenzproblem) reduziert.
   - Wäre daher P_2 entscheidbar, so auch P_1
   - Weiss man bereits, dass P_1 unentscheidbar ist, so folgt daher, dass auch P_2 unentscheidbar ist.

Formaler betrachten wir (o.B.d.A.) einstellige Relationen $R \subseteq \Sigma^*$

3.15.8 Definition 5.11 (Reduktion)

1) Eine Reduktion von $R_1 \subseteq \Sigma^*$ auf $R_2 \subseteq \Sigma^*$ ist eine berechenbare Funktion

$$f : \Sigma^* \to \Sigma^*$$

für die gilt:

$$w \in R_1 \underline{\text{ gdw. }} f(w) \in R_2$$

2) Wir schreiben

$$R_1 \leq_m R_2 \qquad\qquad R_1 \text{wird auf } R_2 \text{ reduziert}$$

falls es eine Reduktion von R_1 nach R_2 gibt.

3.15.9 Lemma 5.12

1) $R_1 \leq_m R_2$ und R_2 entscheidbar $\Rightarrow$ R_1 entscheidbar

2) $R_1 \leq_m R_2$ und R_1 unentscheidbar $\Rightarrow$ R_2 unentscheidbar

Beweis

1) Um $w \in R_1$ zu entscheiden
   - berechnet man $f(w)$ und
   - entscheidet $f(w) in R_2$

2) Folgt unmittelbar aus 1)

Mit Hilfe einer Reduktion des Halteproblems kann man zeigen: Jede nichttriviale semantische Eigenschaft von Programmen (DTM) ist unentscheidbarl

- Semantisch: heißt hier: Die Eigenschaft hängt nicht von der syntaktischen Form des Programms, sondern nur von der berechneten Funktion ab.
- Nichttrivial: Es gibt berechenbare Funktionen, die sie erfüllen, aber nicht alle berechenbaren Funktionen erfüllen sie.

Beispiele für solche Eigenschaften:

- die berechnete Funktion ist total, d.h. die DTM hält für jede Eingabe.
- die berechnete Funktion ist bei Eingabe ϵ definiert (Halteproblem).
- die berechnete Funktion ist die Nullfuntion, d.h. der Funktionswert ist für jede Eingabe gleich 0.
- die berechnete Funktion liefert bei Eingabe ϵ den Wert 0.
- ...

3.15.10 Satz 5.13 (Satz von Rice)

Es sei E eine Eigenschaft partiell berechenbarer Funktionen $f : \Sigma^* \to \Sigma^*$ so dass gilt:

$$\emptyset \subset \{f : \Sigma^* \to \Sigma^* | f \text{ erfüllt } E\} \subset \{f : \Sigma^* \to \Sigma^* | f \text{ ist partiell berechenbar}\}$$

dann ist

$$L(E) := \{code(A) | \text{die von } A \text{ berechnete Funktion } f : \Sigma^* \to \Sigma^* \text{ erfüllt } E\}$$

unentscheidbar.

Beweis

Angenommen $L(E)$ ist entscheidbar
Wir zeigen, dass man dann auch ein Entscheidungsverfahren für das Halteproblem erhält (Widerspruch zu Satz 5.7).
Mit f_u bezeichnen wir die überall undefinierte Funktion, welche offenbar partiell berechenbar ist. O.B.d.A. erfülle f_u die Eigenschaft E.
Sonst könnte man statt E die Eigenschaft $\overline{E}$: "f erfüllt E nicht"betrachten. mit $L(E)$ ist auch $L(\overline{E}) = \overline{L(E)}$ entscheidbar.
Da nicht alle partiell berechenbaren Funktionen E erfüllen, gibt es eine partiell berechenbare funktion g, welche E nicht erfüllt.
Es sei nun A_g eine DTM für g und A_u eine für f_u. Wir konstruieren nun eine DTM A', welche eine Funktion $f : \Sigma^* \to \Sigma^*$ berechnet mit:

$$w = code(a) \text{ für eine DTM , die auf leerer Eingabe terminiert}$$
$$\text{gdw } f(w) \notin L(E) \qquad (*)$$

Wäre daher $L(E)$ entscheidbar, so auch das Halteproblem.

Konstruktion von A':
Die DTM A' testet bei Eingabe w zunächst, ob w Kodierung einer DTM ist.

- Falls nein, so gibt $A' code(A_u) \in L(E)$ aus
- Falls ja, so ist $w = code(A)$ für eine DTM A. Die DTM A' gibt dann $code(A'')$ aus, wobei A'' noch geeignet zu definieren ist:

$$code(A'') \notin L(E) \text{ gdw. } A \text{ hält auf leerer Eingabe}$$

Definition von A'':

- A'' ignoriert zunächst die Eingabe x und simuliert das Verhalten von A auf dem leeren Eingabeband.

- Im Anschluss (d.h. falls A auf leerem Eingabeband terminiert) verhält sich A'' wie A_z bei Eingabe x.

Damit gilt für A'' offenbar:

- Terminiert A auf leerer Eingabe nicht, so berechnet A'' die Funktion f_u, d.h. $code(A'') \in L(E)$, da f_u E erfüllt.
- Terminiert A auf leerer Eingabe, so berechnet A'' die Funktion g, d.h. $code(A'') \notin L(E)$, da g E nicht erfüllt.

Insgesamt haben wir also gezeigt, dass die von A' berechnete Funktion f die Bedingung (*) erfüllt, das Halteproblem also auf das Problem, $L(E)$ zu entscheiden, reduziert.
Da das Halteproblem unentscheidbar ist, folgt die Unentscheidbarkeit von $L(E)$.

3.16

3.16.1 Definition 6.1 (Postsches Korrespondenzproblem)

Eine Instanz es Postschen Korrespondenzproblems (PKP) ist gegeben durch eine endliche Folge

$$P = (x_1, y_1), \ldots, (x_k, y_k)$$

von Wortpaaren mit $x_i, y_i \in \Sigma^+$ für ein endliches Alphabet Σ.
Eine Lösung des Problems ist eine Indexfolge $i_1, \ldots, i_m$ mit

- $m > 0$ und
- $i_j \in \{1, \ldots, k\}$,

so dass gilt: $x_{i_1} \ldots x_{i_m} = y_{i_1} \ldots y_{i_m}$.

Beispiel

siehe Skript 48

Bemerkung

Um die Unentscheidbarkeit des PKP zu zeigen, führen wir zunächst ein Zwischenproblem ein, das modifizierte PKP (MPKP):
Hier muss für die Lösung zusätzlich $i_1 = 1$ gelten, d.h. das Wortpaar, mit dem man beginnen muss ist festgelegt.

3.16.2 Lemma 6.3

Das MPKP kann auf das PKP reduziert werden.

Beweis

Siehe Vorlesung 49
$\rightarrow$ Das MPKP P hat eine Lösung gdw. Das PKP f(P) hat eine Lösung

Beispiel

$P = (a, aaa), (aab, b)$ ist als MPKP lösbar mit Lösung 1,2.

$$f(P) = (\#a\#, \#a\#a\#a), (a\#, \#a\#a\#a), (a\#a\#b\#, \#b), (\$, \#\$)$$

Die Lösung 1, 2 von P liefert die Lösung 0,2,3 von $f(P)$:

$$\#a\#|a\#a\#b\#|\$$$
$$\#a\#a\#a|\#b|\#\$$$

Offenbar muss jede Lösung von $f(P)$ mit 0 beginnen.
Wäre daher das PKP entscheidbar, so auch das MPKP. Um die Unentscheidbarkeits des PKP zu zeigen, genügt es also zu zeigen, dass MPKP unentscheidbar ist.

3.16.3 Lemma 6.4

Das Halteproblem kann auf das MPKP reduziert werden.

Beweis

siehe Vorlesung 50 ff.

3.16.4 Satz 6.5

Das PKP ist unentscheidbar.
Wir verwenden dieses Resultat um Unentscheidbarkeit von Porblemen für kontextfreie und kontextsensitive Sprachen nachzuweisen. Wir zeigen zunächst:

3.16.5 Lemma 6.6

Es ist nicht entscheidbar, ob für kontextfreie Grammatiken G_1, G_2 gilt:

$$L(G_1) \cap L(G_2) \neq \emptyset$$

3.16.6 Beweis

siehe Vorlesung 52.

Kapitel 4

Übung

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

4.1 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:
$X(\widehat{\sigma\theta}) = (X\hat{\sigma})\hat{\theta}$

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

Fallunterscheidung siehe Lehrbuch

b) Strukturelle Induktion siehe Lehrbuch

- IA

4.3 Semantik

4.3.1 Beispiele zur Interpretationsanwendung

a) (1) $f(g(f(a, g(a))), a)^{I_1}$

$$\begin{aligned}
\rightarrow & f^I((g(f(a, g(a)))^I, (a)^I) \\
& = f^I(g^I(f^I(a^I, (g(a))^I)), 0) \\
& = f^I(g^I(f^I(0, s(0))), 0) \\
& = f^I(g^I(s(0)), 0) \\
& = s(s(0))
\end{aligned}$$

(2) $f(g(a), f(a, a))$ äquiv. zu a)

(3) $[(\forall X)(p(X) \to \neg q(q(X)))]^{I,Z}$

Sei Z eine beliebige Variablenzuordnung bzgl. I_1

$\to = \top$ gdw

$\forall d \in D : [(p(X) \to \neg q(g(X))]^{I,\{X \mapsto d\}Z} = \top$ gdw

$\forall d \in D : [p(X)]^{I,\{X \mapsto d\}Z} \to^* [\neg q(g(X)]^{I,\{X \mapsto d\}Z} = \top$ gdw

$X^{I,\{X \mapsto d\}Z} \in p^I$ impl $[\neg q(g(X))]^{I,\{X \mapsto d\}Z} = \top$ gdw

Angenommen $d \in D$ und es gelte $X^{I,\{X \mapsto d\}Z} \in p^I$:

z.Z.: $[\neg q(g(X))]^{I,\{X \mapsto d\}Z} = \top$

$[X]^{I,\{X \mapsto d\}Z} = X^{I,\{X \mapsto d\}Z} = d$

$p^I = \{0\}$, dann muss $d = 0$:

$[\neg q(g(X))]^{I,\{X \mapsto d\}Z} = \top$ gdw

$\neg^* [q(g(X))]^{I,\{X \mapsto d\}Z} = \top$ gdw

$[q(g(X))]^{I,\{X \mapsto d\}Z} = \bot$ gdw

$[q(X)]^{I,\{X \mapsto d\}Z} \notin q^I$

$= g^I(X^{I,\{X \mapsto d\}Z}) \notin q^I$

$= s(d) \notin q^I$

$\Rightarrow$ Offensichtlich falsch, da $q^I = D$

b) (1) Bestimmen Sie den Wert von $g(f(a, g(b)))$ unter der Interpretation I_2

(2) Geben Sie einen Term t aus ... an

(3) Bestimmen Sie (Formelauswertung)

4.3.2 Verschiedene Interpretationen einer Formel

a) Interpretation betrachten

b) Geben Sie eine Interpretation I_2 an unter ...

c) Zeigen Sie, dass die Formel F nicht allgemeingültig ist

4.3.3 Existenz einer Herbrand-Interpretation

$I = (D, \circ^I), D = T(F)$ ist nicht leer, weil wir voraussetzen, dass minestens
1 Konstantensymbol in unserer Sprache enthalten ist. Jedem n-stelligen Funktionssymbol
f weisen wir folgende Funktion zu:
$f^I(t_1, \ldots, t_n) \mapsto f(t_1, \ldots, t_n)$
$p^I = \emptyset$
z.Z.: $\forall t \in T(F) : t^I = t$

Beweis über strukturelle Induktion:

IA

$$a \in T(F) \text{ Konstantensymbol}$$
$$\rightarrow a^I = a$$

IH

Die Aussage gelte $t_1, \ldots, t_n \in T(F)$

IS

$$\forall f(t_1, \ldots, t_n) \in T(F) :$$
$$[f(t_1, \ldots, t_n)]^I$$
$$= f^I(t_1^I, \ldots, t_n^I)$$
$$=^{IH} f^I(t_1, \ldots, t_n)$$
$$=^{DefI} f(t_1, \ldots, t_n)$$

4.4 Äquivalenz und Normalenform

fill

4.5 Unifikation

fill

4.6 Beweisverfahren

4.6.1 Resolutionsverfahren

a) $< [p(X,Y), q(a,Y)], [\neg p(b,a)], [\neg q(Z,V)] >$

$[X,Y], q(a,Y)]$	(1)
$[\neg p(b,a)]$	(2)
$[\neg q(Z,V)]$	(3)
$res(1,2)$	(4)
$res(3,4) \rightarrow []$	(5)

b)

$[q(f(a), f(b))]$	(1)
$[\neg p(X,Y), \neg p(f(a), g(X,b)), \neg(X,Z)]$	(2)
$[p(f(X), g(Y,b)), \neg q(Y, f(Y))]$	(3)
$[\neg p(X_1,Y_1), \neg p(f(a), g(X_1,b)), \neg(X_1,Z_1)]$	$(2')$
$[p(f(X_2), g(Y_2,b)), \neg q(Y_2, f(Y_2))]$	$(3')$
Faktor. von 2' $\sigma\{X_1 \mapsto f(a), Y_1 \mapsto g(f(a),b)\}$:	
$[\neg p(f(a), g(f(a),b), \neg q(f(a), Z_1)]$	(4)
$[\neg p(f(a), g(f(a),b), \neg q(f(a), Z_4)]$	$(4')$
res(1',4') $[\neg p(f(a), g(f(a),b))]$	(5)
$[(q(f(a)), f(Y_5)]$	$(1'')$
res(3',1") $\sigma = \{Y_2 \mapsto f(a), Y_5 \mapsto f(a)\}$	
$[p(f(X_2), g(f(a),b))]$	(6)
$[\neg p(f(a), g(f(a),b))]$	$(5')$
$p(f(X_6), g(f(a),b))]$	$(6')$
res(5',6') $\sigma = \{X_6 \mapsto a\}$	
$[]$	(7)

4.6.2 Schrittweiser Resolutionsbeweis

$(\exists Y)(\forall U)(\neg(\forall U)q(U,Y) \vee q(f(Y),U))$

1 Negation $\neg F$

2 Variablen auseinander divideren

$$\neg(\exists Y)(\forall U)(\neg(\forall Z)q(Z,Y) \vee q(f(Y),U))$$

3 Pränex-Normal-Form bilden

$$
\begin{aligned}
&(\forall Y)\neg(\forall U)F_1\\
&(\forall Y)(\exists U)\neg F_1\\
&(\forall Y)(\exists U)\neg(\neg(\forall Z)q(Z,Y)\vee q(F(Y),U))\\
&(\forall Y)(\exists U)\neg((\exists Z)\neg q(Z,Y)\vee q(f(Y),U))\\
&(\forall Y)(\exists U)\neg(\exists Z)(\neg q(Z,Y)\vee q(f(Y),U))\\
&(\forall Y)(\exists U)(\forall Z)\neg(\neg q(Z,Y)\vee q(f(Y),U))
\end{aligned}
$$

4 Skolem-Normal-Form bilden

$$
\begin{aligned}
&(\forall Y)\sigma=\{U\mapsto g(Y)\}\\
&\rightsquigarrow(\forall Y)(\forall Z)\neg(\neg q(Z,Y)\vee q(f(Y),g(Y)))
\end{aligned}
$$

5 Klauselform bilden

$$
\begin{aligned}
&\forall<[\neg(\neg q(Z,Y)\vee q(f(Y),g(Y))]>\\
&<[\neg\neg q(Z,Y)],[\neg q(f(Y,g(Y))]>\\
&<[q(Z,Y)],[\neg q(f(Y),g(Y))]>
\end{aligned}
$$

6 Resolutionsverfahren

$$
\begin{aligned}
&[q(Z_1,Y_1)] && (1')\\
&[\neg q(f(Y_2),g(Y_2))] && (2')\\
\text{res(1',2')}\ &[] && (3)
\end{aligned}
$$

4.6.3 Notwendigkeit der Faktorisierung

$$
\begin{aligned}
F=<&[p(),p()],[\neg p(),\neg p()]>\\
\text{res 1,2)}\ &[p(),\neg p()] && (3)\\
\text{fakt 1'}\ &[p] && (4)\\
\text{fakt 2'}\ &[\neg p] && (5)\\
\text{res 4',5')}\ &[] && (6)
\end{aligned}
$$

Beweis über Induktion

4.7 Eigenschaften

4.7.1 Beispiel für korrespondierendes Herbrand-Modell

$$
\begin{aligned}
D^I &= \mathbb{N} \\
a^I &= 0 \\
s^I &= x \mapsto x+1 \\
p^I &= \{(x,y,z) | x \leq y+z\}
\end{aligned}
$$

Herbrand Modell

$$
\begin{aligned}
D^J &= T(F) \text{ Menge von Termen über F} \\
&= \{a, s(a), s(s(a)), \ldots\} \\
p^J &= \{(s^x(a), s^y(a), s^z(a)) | x \leq y+z\}
\end{aligned}
$$

4.8 Nachtrag Logik

$$F \vDash p(t) \text{ für alle t} \in T(F) \rightarrow F \vDash (\forall X) p(X)?$$

Nein. Hier: häufige Fehler!! Gegenbeispiel:

$L(R,F,V)$ bestimmen mit$F = \{c_1, \ldots, c_n\}$
Sei G Formel:
$G = p(c_1) \wedge p(c_2) \wedge \ldots \wedge p(c_n)$
Wiederspruch ("Missmatch"mit Term und Domäne) mit:
$D = \{1, \ldots, n+1\}$
$c_1^I = \{1, \ldots, c_n^I = n\}$
$p^I = \{1, \ldots, n\}$

4.9 Turing-Maschine

4.9.1 Palindrom Turingmaschine

$$
\begin{aligned}
A =& (Q, \{a,b\}, \{a,b,\not b,\}, q_0, \Delta, F)\\
\Delta =& \{(q_0, a, \not b, r, q_2) && (q_0, b, \not b, r, q_3)\\
& (q_2, \not b, \not b, n, q_4) && (q_3, \not b, notb, n, q_4)\\
& (q_2, *, *, r, q_a) && (q_2, *, *, r, q_b)\\
& (q_a, *, *, r, q_a) && (q_b, *, *, r, q_b)\\
& (q_a, \not b, \not b, l, q_{av}) && (q_b, \not b, \not b, l, q_{b,v})\\
& (q_{av}, a, \not b, l, q_{l1}) && (q_{bv}, b, \not b, l, q_{l1})\\
& (q_l, x, x, l, q_l) && x \in \{a,b\}\\
& (q_{l1}, \not b, \not b, n, q_F) && (q_l, x, x, l, q_l)\\
& (q_l, \not b, \not b, r, q_0) && \}
\end{aligned}
$$

4.9.2 Wort 2-Band Turingmaschine

$$
\begin{aligned}
q_0 :& \underline{a}ba && \underline{\not b}\ \not b\ \not b\\
q_0 :& a\underline{b}a && a\underline{\not b}\ \not b\\
q_0 :& ab\underline{a} && ab\underline{\not b}\\
q_0 :& aba\underline{\not b} && aba\underline{\not b}\\
q_1 :& aba\underline{\not b} && ab\underline{a}\\
q_1 :& aba\underline{\not b} && a\underline{b}a\\
q_1 :& aba\underline{\not b} && \underline{a}ba\\
q_1 :& aba\underline{\not b} && \underline{\not b}aba\\
q_2 :& aba\underline{\not b} && \underline{a}ba\\
q_2 :& abaa\underline{\not b} && \not b\underline{b}a\\
q_2 :& abaaba\underline{\not b} && \not b\ \not b\ \not b\underline{\not b}
\end{aligned}
$$

4.10

4.10.1 Berechenbarkeit, Entscheidbarkeit, Aufzählbarkeit

a) wahr, nach Def.

b) wahr, gilt nach Satz 2.4

c) wahr, Beweis durch Gegenannahme falsch

d) falsch: sonst partiell entscheidbar = entscheidbar

4.10.2 Primitiv rekursive Funktionen

a)

- PROD:

$$
\begin{aligned}
&f : \mathbb{N}^{n+1} \mapsto \mathbb{N} \\
&g : \mathbb{N}^{n} \mapsto \mathbb{N} \\
&h : \mathbb{N}^{n+1} \mapsto \mathbb{N} \\
&\text{falls gilt:} \\
&f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n) \\
&f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, f(x_1, \ldots, x_n, y), y) \\
PROD(X, 0) &= 0 = g(X) = null^{(1)}(X) \\
PROD(X, Y+1) &= PROD(X, Y) + Y = h(X, PROD(X, Y), Y) \\
&= add(\pi_2^{(3)}(X, PROD(X, Y), Y), \pi_3^{(3)}(X, PROD(X, Y), Y) \\
&= add(\pi_2^{(3)}(X, PROD(X, Y), Y), \pi_3^{(3)}(X, PROD(X, Y), Y))
\end{aligned}
$$

- MODDIFF:

$$
\begin{aligned}
MD(X, 0) &= X = g(X) = id_{(1)}^{(1)} \\
MD(X, Y+1) &= MD(X, Y) - 1 \\
min1(X) &= \begin{cases} x - 1 & x > 0 \\ 0 & \text{sonst} \end{cases} \\
&\text{oder:} \\
vor(0) &= 0 = null^{(0)} \\
vor(X+1) &= X = h(vor(X), X) \\
&\text{dann:} \\
MD(X, Y+1) &= vor(\pi_2^3(X, MD(X, Y), Y))
\end{aligned}
$$

b)

- f_1

$$
\begin{aligned}
sum(0) &= \Sigma_{i=0}^{0} = 0 = g = null^{(0)} \\
sum(y+1) &= \Sigma_{i=0}^{y+1} = add(sum(Y), Y+1) \\
&= h(sum(Y), Y) \\
&= add(\pi_1^{(2)}(sum(Y), Y), s(\pi_2^{(2)}(sum(Y), Y))) \\
f_1(x_1, \ldots, x_k, x_{k+1}) &= sum(g(x_1, \ldots, x_k, i)) \\
f_1(x_1, \ldots, x_k, 0) &= g(x_1, \ldots, x_k, 0) \\
f_1(x_1, \ldots, x_k, Y+1) &= add(f_1(x_1, \ldots, x_k, y), g(x_1, \ldots, x_k, Y+1)) \\
&= h(x_1, \ldots, x_k, f_1(x_1, \ldots, x_k, Y), Y) \\
&= add\Big(\pi_{k+1}^{(k+2)} \underbrace{(x_1, \ldots, x_k, f_1(x_1, \ldots, x_k, Y), Y)}_{=l}, \\
&\qquad g\big(\pi_1^{(k+2)}(l), \ldots, \pi_k^{(k+2)}(l), s(\pi_{k+2}^{(k+2)}(l))\big)\Big)
\end{aligned}
$$

tadaaaaa!!!

- f_2 sind wir nichmehr zu gekommen

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

5.5 fill 03.05.16

5.6

5.6.1 Intel Itanium

Intel Itanium 2 1.5GHz, Fortran/C
Tabelle siehe Folien

Quad-Core Itanium

- Bereiche mit unterschiedlicher Taktrfrequenz und Spannung
- 6MB L3 pro Kern
- QPI-Links zur Verbindung mit anderen Prozessoren (Statt FSB)
- On-chip Speichercontroller

8-core Itanium

- Gemeinsamer L3 Cache für alle Kerne
- Neue Mikroarchitektur für höhere IPC
  - Fetch und Decode: 6 Instruktionen pro Takt (2 Befehls-Gruppen)
  - Execution: 12 Instruktionen pro Takt (unabhängig von Gruppen)

5.6.2 IBM Power

Die Power Architektur

POWER: Performance Optimization With Enhanced RISC

90 IBM POWER1

93 IBM POWER2

96 IBM POWER2 Superchip (P2SC)

93 Einführung der PowerPC-Prozessoren

98 Einführung der IBM POWER3-Prozessoren

01 Einführung des IBM POWER4-Prozessors

04 POWER5

07 POWER6

10 POWER7

14 POWER8

POWER4+ Processor

- 2 Cores pro Chip
- Pro Core l1 D-Cache und L1-l-Cache
- Shared L2-Cache
- L3-Cache off chip
- 2 FPUs pro Core
- Fused-Multiply-Add
- 6,8 GFLOPS@1,7GHz

Speicher IO-Bandbreite

Grafik siehe Folien

8Prozessor SMP System(MCM: 11x11cm

Grafik siehe FOlien

Mikroarchitektur des Proessors IBM POWER4

- Jeder iBM POWER4-Chip besitzt zwei 64-Bit-Mikroprozessorkerne
- Der gemeinsam genutzte L2-Cache ist in 4 gleichgroße Teile mit jeweils einem speraten L2-Cache-Controller aufgeteilt
- Das L3-Cache-Directory und der L3-Cache-Controller befindensich auf dem Chip während der L3-Cache selbst aus DRAM besteht
- Der für die I/O-Operationen verantwortliche GX-Controller ist mit zwei, je einer Richtung zugeordneten 4-Byte-breiten GX-Bussen ausgestattet
- Fabric-Controller
- Der Mikroprozessorkern ist n Form einer typisch superskalaren Mikroarchitektur implementiert
- 8 unahängige Funktionseinheiten, die nebenläufig parallel arbeiten können
- Jede der FPUs ist in der Lage einen Multiply/Add-Befehl pro Takt fertigzustellen

Befehlsausführung und Ressourcnverwaltung

- Der IBM POWER4 arbeitet mit spekulativer Befehlsausführund, d.h. Befehle werden ausgeführt bevor sicher ist, ob sie überhaupt benötigt werden
- Zur Vermeidung von Pseudoabhängigkeiten durch out-of-order Processing wird mit Register-Renaming gearbeitet

Pipeline

Grafik siehe Folien

Instruction Fetch, Group Formation und Dispatch 1

- Die Adresse für den L1-I-Cache-Zugriff steht im Insturction Fetch Adress Register. Die adressinformation erhält das IFAR von der BR-Logik
- In der IF-Phase werden die Befehle aus dem L1-Cache geholt und in die Instruction-Queue geladen. Die I-Queue ermöglicht dabei die Fortsetzung des Pipelinings auch bei einem Cache-Miss
- Die geholten Befehle werden nach Sprugbefehlen durchsucht
- In den Phasen Dß bis GD werden die Befehle decodert, zerlegt, zu gruppen formatiert
- ...

Out-of-Order Processing

Load/stor Istruction Processing

Floating-Point Execution Pipeline

Group Completion

- Eine Befehlsgruppe wird beendet, wenn die Ergebnisse in das Zielregister geschrieben wurde und die Ergebnisse somit für das weitere Programm verfügbar sind.
- Wurde ein Befehl oder eine Befehlsgruppe spekulativ ausgeführt kann sie erst vollendet werden, falls alle Bedingungen für seine Ausführung erfolgreich waren
- Eine Befehlsgruppe kann erst vollendet werden, wenn alle älteren Gruppen komplett und korrekt abgeschlossen wurden
- Por Takt ist eine Befehlsgruppe komplettierbar

Matrix Multiplication

Grafik siehe Folien

5.6.3 POWER7

- 32MB shared L3-Cache; eDRAM
- unterschiedliche SRAM Zellen für L1- und L2-Cache
- 4 Threads pro Kern
- Local (MCM) und Remote SMP Interface

SMP System

- 4 Chips pro MCM
- 8 MCMs können direkt verbunden werden
  - 256 Kerne; 1024 Threads

POWER Prozessorfamilie

Grafik siehe Folien

5.6.4 MIPS

MIPS Prozessoren

81 MIPS RISC ISA gegründet

84 Gründung MIPS

85 erster kommerzieller RISC-CHIP: MIPS R2000

88 R3000

91 R4000

94 R8000

96 R5000

96 R10000

98 R12000

02 R16000

R10k

- Ab dem Befehlsatz MIPS III werden folgende Datenregister auf ISA-Ebene unterstützt
  - 32 General Purpose Register zu je 64Bit
  - 32 FLoating Point Register zu je 64Bit
- Ab MIPS I werden folgende Adressierungsarten unterstützt:
  - Direktwert-A
  - Direkte Register-A
  - PC-relative A
  - Relative Hauptspeicheradresssierung über ein Register mit Displacement
- Ab MIPS IV gib es im beschränkten Umfang indexierte A

Befehlssätze der Prozessoren R10k und R12k

ALU-Befehle

Mikroarchitektur der Prozessoren R10k und R12k

Pipelines des MIPS R10k

- Der superskalare R10k arbeitet mit 5 weitestgehend unabhängigen Pipelines und ebso vielen Ausführungseinheiten
- Für jeden Operationstyp steht eine Warteschlange für Befehle zur Verfügung
- Die Load/Store-Warteschlange ist FIFO-georndet, um Speicherabhängigkeiten und Datenkonsistenz zu gewährleisten
- Die anderen beiden Warteschlangen sind keiner strengen Ordnung unterworfen

Pipeline-Stufen 3 bis 7

Floating Point Units

- Floating-Point-Multiplizierer
- Floating-Point-Dividierer und Radizierer
- Floating-Point Addierer

Integer Units und Load/Store Unit

- Integer-Einheit 1
- Integer-Einheit 2
- Load/Store

Godson-3

- Basis für chinesisches Supercomputer Programm
- 4-fach superskalar mit out-of-order Execution
- MIPS64 ISA mit Erwetierung für x86 Emulation
- Quad-Core Design
  - On-chip Speichercontroller
  - HyperTransprot zur Kommunikation mit chipsatz und anderen Prozessoren

5.6.5 SPARC

82 Wesentliche Forschungsarbeiten an der University of California Berkeley auf dem Gebiet von RISC-Prozesorarchitekturen

87 Sun Microsystems stellt die SPARC-Architektur vor: Scalable Processor Architecture

90 erste 32-Bit Implementierung

95 SPARC64/UltraSparc mit 64-Bit Verarbeitungsbreite

05 UltraSparc T1

07 UltraSpard T2

09 Sparc64 VIIIfx

10 Sparc T3: 16 Kerne, 8fach SMT

SUN Niagara II (T2)

- 8 Prozessorkerne
- 4 MB L2-Cache
- Max. 16 DDR2 FB-DIMMs
- 2-mal 10Gb/s Netzwerk
- x8 PCI-Express

T2 Multithreading

- Abwechelndes Arbeiten an mehreren Threads, um Speicherlatenz (Cahcelatenz zu verdecken
  - Kein komplexes out-of-order Design erforderlich
  - Erforder hohes Maß an Parallelität in der Software
- Abwechselndes Arbeiten an mehreren Threads um Speicherlatenz zu verdecken

5.6.6 CELL Broadband Engine

- Heterogener Multi-Core
  - 1 PowerPC Kern (PPE)
  - 8 Synergistic Processing Elemts (SPE)
- SPE:
  - In-order Execution
  - 4 single precision Operationen pro Takt
  - Kein Cache
  - 256 KB Local Store
- Gemeischaftsentwicklung von IBM, Sony und Toshiba
- PowerXCell 8i
- Roadrunner
  - Erster PetaFlop Rechner
  - 3060 hybride Knoten

5.6.7 ARM

- ARM: Advanced RISC Machines
- IP-Cores zur Integration in System-on-Chips
- 32-Bit RISC Architektur
- Zeichnen sich durch geringen Stromverbrauch aus
- Häufige Verzweigungen schlecht für effizientes Pipelining
- Paralleles Ausführen beider Pfade bei if-then-else

- Einsatz: unterschiedliche Kerne mit gleichem Befehlssatz
- Cortes A15 liefert doppelte bis dreifache Rechenleistung des Cortex A7
- Cortex A7 hat drei- bis vierfache Rechenleistung pro Watt

5.6.8 DEC ALPHA

- Alpha 21064 und 21164 in Supercomputern on Cray
- 2004 nach Übernahme durch HP eingestellt
- 64 Bit RISC Architektur

5.7 24.05.2016

fill

5.8 31.05.2016

fill

5.9 07.06.2016

5.9.1 Verbinungsnetzwerke

Ausgewählte Metriken statischer Verbindungsnetzwerke II

- Knotenvernetzung: Minimale Knotenausfallzahl für den zerfall des netzes in zwei (beliebig große) Teile
- Kantenvernetzung: Minimale Kantenausfallzahl für den Zerfall des Netzes in zwei (i.a. verschieden große) Teile
- Halbierungsbreite: Minimale Kantenausfallzahl für den zerfall des netzes in zwei gleichgroße Teile (Spezialfall der Kantenvernetzung)
- Bisektionsbandbreite: Ausgangspunkt: Teilung des Netzes in zwei gleichgroße Teile. Multiplikation der verbleibenden Verbindungskanäle zwischen den Netzhälften mit der Übertragungsbandbreite
- Konnektivität: Minimum aus den Werten Knotenvernetzung und Kantenvernetzung
- Erweiterungsinkrement: Kleinste Knotenzahl für Erweiterung

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

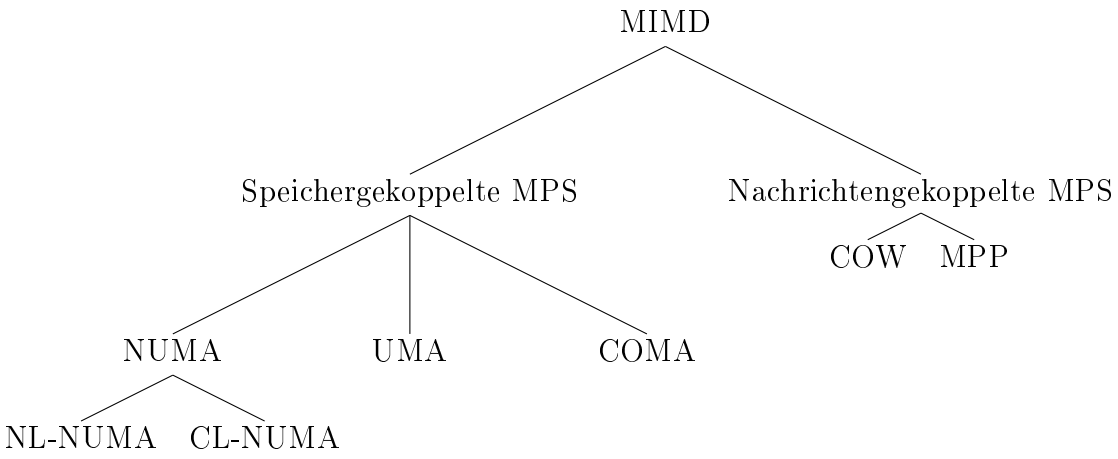
(2) SIMD Single Instruction Stream Multiple Data Streams

- 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

6.2.3 MMX, SSE, AVX

- MMX Mulit Media Extensions
  - 1996 Pentium
  - 64-Bit Register
  - für Integer Verarbeitung
  - Flynn SIMD
- SSE Steuerung SIMD Extensions
  - 1999 Pentium 3
  - SSE 2 2001 Pentium 4
  - MIMX mit eingebunden
  - Flynn SIMD
- Advanced Vector Extensions
  - 2011 Sandy-Bridge
  - 256Bit Register (für Floating-Point-Ops)
  - 8*64Bit
  - 16*32Bit
- AVX 2
  - 256Bit auch für Integer-Befehle
  - Unterstützung FMA (Fused Multiply Add)
- AVX 512
  - Ende 2015
  - 512Bit Register

6.3 Prozessoren / Pipelifting

6.3.1 Adressierungsarten

a) Immediate Operand

- Operand stehlt als Konstante im Befehl:
  z.B. MOVE B R3, hash32
  oder Add R1, hash17 → Addiert den Wert 17 zu Inhalt von R1

b) Immediate Adresse

- Speicheradressierung direkt:
  Effektiver: Adresse steht als absolute Adresse im Befehl
  Operand steht im Speicher:
  z.B.: Add 0xFFC0: hash323

c) Register Direkt

- Adresse steht als kurze Registeradresse im Befehl z.B.: 3Bit für IA 32, Operand steht im Register:
  z.B.: MOVE W SP, R0 transportiert den Wortinhalt von R0 in das Stack-Pointer-Register SP

d) Register Indirekt

- Effektive Adresse steht im Register
- Adressierung erfolgt indirekt über Registerinhalt
- Adressierung wird durch runde Klammern mit der Bedeutung İnhalt von Register"
- z.B.: MOVE H R1,(R0) transportiert einen 16Bit Speicheroperanden; diesen Adressen in R0 steht nach R1

e) Register Indirekt mit Displacement

- Die im Register stehende Adresse wirkt als Basisadresse und wird nicht verändert (steht in Klammer)
- z.B.: MOVE H R2, hash4 (R3)
- transportiert einen 16Bit Speicheroperanden nach R2
- Basisadresse steht in R3
- Adresse des Speicheroperanden = Basisadr. + Displacement (hash4)

6.3.2 scale-potenz

scaled indexed addressing mode
img siehe Übungsblatt

- Basisadresse + Displacement + Index*Scale = Adresse des Operanden
- z.B. Im R1 steht 1000 im R2 steht Index 4:
  MOVE R0, hash8(R1)(R2 * hash4) → mit R0 von Adresse 1024
  MOVE R0, hash1000(R1)(R2 * hash2) → mit R0 von adresse 4000
  MOVE R0, hash1000(R2*hash8) → mit R0 von Adresse 1032
- Weshalb nur eine 2er Potenz für scale?
  → * kein universeller Multiplizierer: mach nur Verschiebung

6.3.3 Speedup

z.B. Intel 80486 Pipeline

- Fetch Instruction FI
- Main instruction Decode D1
  - Befehlslänge einstellen?
  - Operanden decodieren
- Secondary instruction Decode D2
- execution EX
- Write Back WB

	1	2	3	4	5	6	7	8	9
1	IF	D1	D2	EX	WB				
2		IF	D1	D2	EX	WB			
3			IF	D1	D2	EX	WB		
4				IF	D1	D2	EX	WB	
5					IF	D1	D2	EX	WB

k-1 Zeitschritte Einlaufphase
ab k pro Takt 1 Ergebnis (n-Mal)

$$\begin{aligned} T_{Pipe} &= k - 1 + n \\ &= 4 + 30 \\ &= 34 \\ S_{P_{Pipe}} &= \frac{T_1}{T_{P_{Pipe}}} \\ &= \frac{150}{34} \\ &= 4,41 \end{aligned}$$

6.3.4

a)
- IPS : Instruction per Second
  - gibt an wie viele Maschinenbefehle pro Sekunde ausgeführt werden können (Integer, Floating Point, load/store, Sprung-Befehle, ...)
  - Vorteile: einfache Bestimmung da keine Trennung der Befehle notwendig

  - Nachteile: nicht für alles zu gebrauchen z.B. Rechnen arbeitet vorweigend mit Floating Point Operationen double Precision, Einflüsse durch RISC $\leftrightarrow$ CISC: mehr Instruktionen als CISC bei gleichem Programm; Compilerabhängiger Programmablauf, ..., Art der Operationen: 64-Bit $\leftrightarrow$ 16-Bit

- IOPS : Integer Operation Per Second
  - Vorteile: hohe Relevanz für Anwendungen die vorwiegend mit Integer Operationen arbeiten z.B. Bitverarbeitung
  - Nachteile: für Rechnen nicht so interessant, da meistens FP - Operationen
  - Einflüsse: Compiler, Genauigketi und Art der Operation (32Bit Int Div $\leftrightarrow$ 16Bit Integer Div), CISC - RISC

- FLOPS Floating Point Operations per Second
  - ist wichtige Kenngröße für das wissenschaftliche Rechnen
  - Top 500- Liste
    * Auflistung der 500 leistungsfähigsten Computer der Welt
    * Basis der Einordnung ist
      · UNPACK-Benchmark läuft auf jeder Maschine
      · erreichte PFLOPS ist Basis für Einordnung
  - Probleme:
    * hängt stark von der Anwendung ab
    * unterschiedliche Komplexität der Operation wird nicht berücksichtigt (ADD, DIV)
    * für Integer Probleme wie z.B. Bildverarbeitung nicht aussagekräftig

- IOOPS Input/Output Operations per Second
  - werden von Vertreibern von Festplatten, Solid State Disks und SAN (Storage Arial Networks) angegeben
    * z.B.: Intel Solid State Drive 910
      · 800GB
      · IOOPS (Random 4k)- Herstellerangabe
      · Read: 180.000
      · Write 75.000
    * z.B.: Western Digital Velori Raptor
      · 600GB
      · IOOPS (Random 4k)- Messung
      · Read: 140
      · Write 120
  - Benchmarks: IOmeter (Itel), IOzone
  - Unterscheidung:

(1) Random: Positionen zufällig gewählt (kein Streaming)
(2) Sequentiell: z.B. Read lesen von aufeinanderfolgenden Speicherplätzen (Streaming)

– Hinweis: IOOPS werden fälschlicherweise oft als IOPS in der Literatur bezeichnet

b) Für die Bewertung von Allzwecksystemen sollten alle 4 Kenngrößen für eine Reihe von Programmen getestet werden, damit die verschiedenen Nutzen sich einordnen können
→ mit Theoretischen Peak Performance Werten vergleichen

- Peak Performance Werte
  - sind Bestwerte, die anhand der Architektur berechnet werden
  - praktisch werden diese Obergrenzen nicht erreicht
  - z.B.: Top 500 Liste
    * neben Linpack FLOPS wird auch die Theoretische Floating Point Peak Performance angegeben

6.4

6.4.1 SPARC INTEL AMD

a) SPARC64 III

(1) Berechnung der Theoretischen Peak Performance IPS IOPS FLOPS

geg.: $f = 330$ MHz Taktfrequenz
2 Integer Units
2 Floating Point Units
4 Befehle pro Takt
IPS

$$\text{IPS} = f \cdot \text{IPC} = 330 \cdot 10^6 \cdot 4 = 1,32 \text{ GiPS}$$

IPC = Instructions per Cycle(Clock)

$$\text{IOPS} = f \cdot \text{IO-PC} = 330 \cdot 10^6 \cdot 2 = 660 \text{ MIOPS}$$

$$\text{FLOPS} = f \cdot \text{FLO-PC} = 330 \cdot 10^6 \cdot 2 = 660\text{MFLOPS}$$

(2) Wie viele IOOPS?

- Volle Verarbetungsleistung wird erreicht, wenn 2 IOP und 2 FLOP pro Takt ausgeführt werden

- Im worst case (bezogen auf den Speicherzugriff) müssen gleichzeitig Operanden und Befehle aus dem Speicher geholt werden und Ergebnisse zurückgeschrieben werden
  - 4 IOOP für Instruction Fetch
  - 8 IOOP für Operand Fetch
  - 4 IOOP für Write Back
  - → Gesamt: 16 IOOP pro Takt

  Total IOOP $= f \cdot$ IOOP-PC $= 330 \cdot 10^6 \cdot 16 = 5,28$ GIOOPS
  Aktuell nicht Realisierbar in oberen Speicherhierarchieebenen, daher meist Optimierung der Speicherzugriffe durch das Laden mehrerer Instruktionen/-Operanden mit eine I/O -Zugriff

(3) Fused Multiply Add
im eingelaufenen Zustand der Pipeline wird pro Takt eine Multiplikation und eine Addition fertig

- hat das Einfluss auf IPS?
  kein Einfluss
  - mit FMA
    - Multiply/Add ist jetzt eine Instruktion $X = A \cdot B + C$
    - diese Instruction dauert einen Takt
  - ohne FMA
    - Multiply/add sind 2 Instruktionen
    - → dauert 2 Takte
- hat das Einfluss auf FLOPS?
  - FLOPS verdoppeln sich
- hat das Einfluss auf IOPS?
  - bis AVX2 ab Haswell gab es kein FMA für Integer Operationen
  - ab AVX2 auch FMA für Integer → Verdopplung von IOPS

b) INTEL CORE i7-2600K Sandy Bridge

(1) Floating Point Peak Performance (Double Precision)

- 4 Kerne
- 3,4GHz
- pro Kern 5 Instruktionen laden + decodieren
- pro Kern 2 FPUs
- AVX (256Bit)

FPPP = Taktfrequenz * Anzahl der gleichzeitigen FP Operationen * Anzahl der Kerne
$= 3,4 \cdot 10^9 \cdot 4$ (AVX = 64Bit = Double Precision) $\cdot 2$ FPUs $= 108,8$ GFLOPS

(2) Bandbreite = FPPP * Operanden * Operandenbandbreite

$$= 108,8 \cdot 10^9 \text{ FLOPS } \cdot s^{-1} \cdot 3 \text{ Operanden } \cdot 8 \frac{\text{Byte}}{\text{Operand}} \text{ IEEE STANDARD}$$

$$= 2,6 \text{ TB/s} \rightarrow$$

c) AMD FX 8350

(1) theoretische FPPP

geg.: 4 GHz Taktfrequenz
8 Integer Kerne
2 Integer Kerne teilen sich eine FPU
Fused Multiply Add
2 Pipelines pro FPU, jede Pipeline 128 Bit

$$\begin{aligned} \text{FPPP} &= \text{Taktfrequenz } \cdot \text{ Anzahl gleichzeitiger FP Operationen} \\ &= 4 \cdot 10^9 s^{-1} \cdot 2 \text{ (Anzahl der Pipelines) } \cdot 2(128\text{Bit}) \cdot 2 \text{ (FMA)} \\ &= 32 \text{ GFLOPS} \end{aligned}$$

(2) FPPP für ganzen Prozessor

$$\begin{aligned} \text{FPPP} &= \text{FPPP einer FPU } \cdot \text{ Anzahl der FPUs} \\ &= 32 \cdot 4 = 128 \text{ GFLOPS} \end{aligned}$$

6.5

fill 24.05.

6.6 HND, RISC, DLX-Architektur, Blatt 3

6.6.1 Hauptkomponenten HDN (3.1)

Welche Hauptkomponenten nutzt HDN für die Beschreibung von Befehlssätzen?

$\leftarrow$ Transfer logisch (z.B. GPR[R6], Generel Purpose Register)

M Speicherzugriff

$\leftarrow_n$ Transfer mit expliziter Längenangabe

Verkettungsoperator

- Beispiel: GPR[R4] $\leftarrow_{32} M$[Adresse x] ## M[x+2]

X^m Wiederholungsspezifikation

$X_{m...n}$ Zugriff auf eine Bitkette

- Beispiel: $\text{GPR[R6]}|_{0...23} \leftarrow O^{24}$

X_n Zugriff zum Einzelbit

- Beispiel: $\text{GPR[R5]}_0 \ldots 23 \leftarrow M(\text{[Adresse x]})_0^{24}$

6.6.2 RISC-Architekturen (3.2)

Nennen Sie wesentliche Merkmale von RISC-Architekturen.

- wenige Befehlsformate fester Länge (für DIX-Architekturen 32 Bit)
  - erleichtert Pipelining erheblich. Durch konstante Befehlslänge kann schon der nächste Befehl geholt werden. (IF Instruction Fetch) während der vorhergehende Befehl dekodiert wird.
- Ein-Zyklus Operationen
  - im eingelaufenen Zustand der Pipeline stellt jede Verarbeitungseinheit pro Takt einen Befehl fertig
- Load/Store-GPR-Architektur
  - Verarbeitungsbefehle greifen nur zu Universalregistern (GPR General Purpose Register) zu. Damit können nur Load/Store-Befehle zum Speicher zugreifen
- Festverdrahtete Steuerung
  - Vorteil: geht schneller
  - Nachteil: bei Änderungen in der Architektur muss neue festverdrahtete Steuerung erstellt werden

  $\leftrightarrow$ Bei CISC Steuereung der Befehlsabarbeitung durch ein Mikroprogramm
- Reduzierung der Prozessorhardware führt im Vergleich zu CISC zu kürzeren Entwicklungszeiten
  - weniger Chipfläche

6.6.3 Beschränkungen bei DLX

I-Typ-Befehlsformat

- Befehlswort wird in das IR (Instruction Register) geladen
- $^{IR0}Opcode^{IR|IR6}rs1^{IR10|IR11}rs2^{IR15|IR16}Immidiate^{IR31}$

Welche Beschräkungen ergeben sich aus einer einheitlichen Befehlslänge von 32 Bit bei der DLX-Architektur?

1 Bei den Direktoperanden sind wir auf 16 Bit begrenzt, obwohl wir eine Registerlänge von 32 Bit haben

   - Nutzung von 2 Befehlen, um ein Register mit einem 32 Bit Direktoperanden zu laden
   - LHI R10, 0x1234; lade höheres Halbwort mit Immediate 16;
     R10 $\leftarrow$ 0x1234 ## 0^{16}
   - ADDUI R10, R10, 0x8678; Addiere unsigend immediate; unsigned verwendet automatisch Null-Erweiterung

$$\begin{aligned} &12340000\\ +&00008678\\ =&12348678 \end{aligned}$$

2 nur 6 Bit für Operationscode $\rightarrow 2^6 = 64$Befehle

3 nur 5 Bit für Kodierung des Registers $\rightarrow 2^5 = 32$ Register Kodierbar

$2 \leftrightarrow 3$ mehr Register z.B.: $2^6 = 64$ bedeutet weniger Befehle $2^4 \rightarrow 16$ Befehle

6.6.4 Zuordnung unter HDN

Was versteht man unter einer vorzeichenerweiterten Zuordnung zu einem 32-Bit-Register und wie kann diese in HDN dargestellt werden?

- vorzeichenerweiterte Zuordnung
   - Problem: Direktoperand hat nur 16 Bit IR16...31
     * wenn eine Operation mit einem 32-Bit-Register erfolgen soll muss der Direktoperand auf 32 Bit erweitert werden, ohne dass sich der Wert des Direktoperanden verändert
   - Lösung: Das MSB (Most Significant Bit) IR_16 muss 16 mal vor den Direktoperanden verkettet werden:
     $(IR_{16})\#\#IR_{16...31}$
     a) $IR_{16} = 0 \rightarrow$ positive Zahl
     b) $IR_{16} = 1 \rightarrow$ negative Zahl (Zweierkomplement)

6.6.5 HDN Beschreibung I-Typ-DLX

Interpretieren Sie die HDN-Beschreibung des I-Typ-DLX-Befehlsformates!

- Immediate-ALU-Befehle
  - GPR[rd] $\leftarrow$ GPR[rs1]op($(IR_{16})^{16}\#\#IR_{16\ldots31}$
  - Direktoperand ($IR_{16\ldots32}$ wird vorzeichenerweitert und über der Operationscode op(ADD, SUB, MUL, AND, ...) mit dem Inhalt des GPR, dessen Nummer auf den Bitstellen 6 ... 10 (rs1) steht, verknüpft. Das Ergebnis wird mit dem GPR, dessen Nummer auf den Bitstellen 11... 15 des Befehlsowrtes steht, abgespeichert.
  - Beispiel: ADDI rd,rs_1, 0xF43A; GPR[rd] $\leftarrow$ GPR[rs_1]+0xF43A
- Lade-Speicher-Befehle
  - Lade-Befehle
    - GPR[rd]$\leftarrow$ M[GPR[$rs_1 + ((IR_{16})^{16}\#\#IR_{16\ldots31})$]
    - Beispiel
      a) LW rd, D(rs_1) allgemeine Darstellung
      b) LW R1, 10(R2) ; R! $\leftarrow$ M[10+R2]## M[11+R2] ## M[12+R2] ## M[12+R2]
  - SPeicher-Befehle
    - M(GPR[rs_1]+$((IR_{16})^{16}\#\#IR_{16\ldots31})$]$\leftarrow$GPR($rs_2$
- bedingte Verzweigung
  - if/GPR[rs1] PC $\leftarrow$ PC + $((IR_{16})^{16}\#\#IR_{16\ldots31}$
    Wenn die Sprungbedingung erfüllt ist, wird de rPC um das vorzeichnerweiterte Immediate erhöht und somit gesprungen. Sprungbedingung:
    - BEQZ (Brand EQual Zero) $\rightarrow$ Wert in rs1 = 0
    - BNEQZ (Brand Not EQual Zero) $\rightarrow$ Wert in rs1 != 0
- Sprungbefehl mit Zieladresse im Register
  - rd = 0, $IR_{16\ldots31} = 0$
  - GPR[rs1] = Zieladresse des Sprungbefehls
  - Die durch den PC estimmte Befehlsfolge wird verlassen und bei rs1 fortgesetzt (unbedingter Sprung)

6.6.6 DLX-Architektur

Wo zeigen sich bei der DLX-Architektur besonders die Gestaltungsgrundsätze SSparsamkeitünd Örthogonalität"?

- Sparsamkeit

  → notewendig, da nur wenig Befehls-Kodierungen

  Ziel: möglichst mit einer Befehlskodierung mehrere Aufgaben erfüllen

  Weg: Register R0 hardwaremäßig auf = gesetzt

  – Lade Speicher Befehle

    z.B.: LW rd, D(rs1) allg Darstellung
    → LW R1, 10(R2); R1 ← M[10+R2] ## M[11+R2] ## M[12+R2]## M[13+R2], Register indirekt mit Displacement
    → LW R3, 100(R0) ; R3 ← M[100] ## M[101] ## M[102] ## M[103] , Immediate Adress
    → LW R5,0(R6) ; R5 ← M[R6]##M[1+R6]##M[2+R6]##M[3+R6], Register Indirekt

    z.B.:
    · SW D(rs1), rs2 allg. Darstellung
    · SW 10(R3), R4 ; M[10+R3]##M[11+R3]##M[12+R3]##M[13+R3] ← R4, Register Indirekt mit Displacement
    · SW 100(R0),R4 ; M[100] ## M[101] ## M[102] ## M[103] ← R4, Immediate Adress
    · SW 0(R3), R4 ; M[R3]##M[1+R3]##M[2+R3]##M[3+R3] ← R4, Register Indirekt

    Man kann mit diesem Befehl auch onch den Speicher mit 0 initialisieren
    · SW D(rs1), R0 ; M[D+GPR[rs1]]## M[1+D+GPR[rs1]]##M[2+D+GPR[rs1]]## M[3+D+GPR[rs1]] ← R0

  – R-Typ-Befehlsformat
    z.B.: rd, rs1,rs2 allg. Darstellung
    * Add R3,R2,R1 ; normale Addition R3 ← R2+R1
    * Add R3,R2,R0 ; Registertransfer R3 ← R2
    * Add R3,R0,R0 ; Clear Register R3 ← 0^{32}

- Orthogonalität
  Funktionell unabhängige Teilelemente müssen auch unabhängig voneinander spezifiziert und realisiert sein.
  → spezieller Befehl wird konstruiert durch freie Kombination der Befehlselemente:

  1 Operation
  2 Datentyp
  3 Adressierungsart
  4 Registernummern

6.6.7 DLX-Befehlsfolge mit HDN

Kommentieren sie die folgende DLX-Befehlsfolge mit HDN!

- ORI R5, R0, 0x102C ; R5 $\leftarrow_{32}$ R0 | 0x0000 102C
  danach in R5: 0x0000 102C
- LW R3, 0(R5) ; R3 $\leftarrow_{32}$ M[0+R5]## M[1+R5]## M[2+R5]## M[3+R5]
  ist das Big Endian oder Little Endian? Um was geht es bei Big und Little Endian?
  - Ablage der Daten im Hauptspeicher
  - z.B.: im Regsiter steht 0x12345678
    - Big Endian: Adressen sind Byteweise gespeichert, höherwertigstes byte (hier "12") zuerst
    - Little Endian: Adressen auch hier Byteweise gespeichert, niedrigwertigstest byte (hier "78") zuerst

  $\rightarrow$ Adresse ist als Big Endian abgespeichert
- SUBI R2, R0, 5 ; R2 $\leftarrow_{32}$ R0 - 0x0000 0005
  -5 wird in R2 geladen
- loop: ADDI R2, R2, 1 ; R2 $\leftarrow_{32}$ R2 + 0x0000 0001
  lege Anzahl der Schleifendurchläufe fest $\rightarrow$ 5 Schleifendurchläufe
- SW 0(R3), R4 ; M[0+R3]## M[1+R3]## M[2+R3]## M[3+R3] $\leftarrow$ R4
  Inhalt von R4 wird ab der Speicheradresse gleaden, die in R3 steht
- ADDI R3, R3, 4 ; R3 $\leftarrow_{32}$ R3 + 0x0000 0004
  Inkrementierung der Speicheradresse in R3 um 4, damit in dem nächsten Schleifendurchlauf das nächste Wort (4 Byte) geschrieben werden kann
- BNEZ R2, loop ; Offsetsxt 16 $\leftarrow (IR_{16})^{16}$ ## $IR_{16\ldots31}$
  ; name(loop) $\leftarrow$ PC + Offsetsxt $16_{16\ldots31}$
  ; if(R2 != 0) PC $\leftarrow$ name(loop)
  ;nach 5 Durchläufen sit R0 = 0 $\rightarrow$ dann geht es weiter
- ADD R3, R0, R0 ; löschen von Register R3

fill ...

6.7 fill - Blatt 4

6.7.1 fill

6.7.2

	Grad d	Druchmesser	Konnektivität	Halbierungs-breite	kleinste Er-weiterung
Ring	2	N/2	2	2	1
Vollst. Graph	N-1	1	N-1	$N^2/4$	1
Stern	1 bzw. N-1	2	1	N/2	1
Vollst. Binärbaum	1,2,3	$2 \cdot ld((N+1)/2)$	1	1	N+1
2D-Torus	4	$2\sqrt{N}/2$	4		$2\sqrt{N}+1$
Hypercube	$ld(N)$	$ld(N)$	$ld(N)$	N/2	N

- Ring:
  - einfaches Routing
  - geringe Kosten → geringe Kabelmenge
  - schlechte Skalierung
  - leicht erweiterbar
- Vollständiger Graph
  - hohe Kosten → nur für kleine Netzwerke möglich
  - kein Routing
  - Ausfallsicherheit, schnell
- Stern:
  - leicht erweiterbar
  - schnell
  - viel Kabel in bezug auf Ring, Gitter
  - wenn zentrale station ausfällt bricht Netz zusammen
- vollst. Binärbaum
  - schwachstelle ist Wurzelknoten
  - relativ schwierig das auf Algorithmen abzuleiten → Anwendung für Broadcast
- 2D Torus

  - schnell gegenüber Gitter (halber Durchmesser)
  - konstanter Grad (vorteil)
  - Erweiterbarkeit geht noch (besser als Hyperscale, schlechter als Stern/Ring)
- Hypercube
  - kleiner Durchmesser (ld N)
  - schlechte Erweiterbarkeit

6.7.3 Gitter Topologien

	4x6	3x8	2x12
Grad	2,3,4	2,3,4	2,3
Durchmesser	8	9	12
Halbierungsbreite	4	3	2

6.7.4 Codierung - OMEGA Netz

Routing

- Destination Rooting
  - Bitwertigkeiten ($b_0 b_1 b_2 b_3$) entsprechend den jeweiligen Ausgängen (oberer/unterer) der Betazelle

$$A_6 = 0110 = obA - uA - uA - obA$$

  - Beispiele: $E_1 \rightarrow A_6$ und $E_{14} \rightarrow A_6$
- XOR-Routing
  - XOR-Verknüpfung von Quell und Zieladresse $\rightarrow$ ergibt Schalterstellung der jeweiligen Beta-Zelle (schräg = 1, gerade = 0):

$$E_{14} \rightarrow A_6 \rightarrow 1110 \text{ XOR } 0110 = 1000$$
$$E_1 \rightarrow A_6 \rightarrow 0001 \text{ XOR } 0110 = 0111$$

Blockierung im OMEGA-Netzwerk

- Anzahl der Übereinstimmungen bei Blockierung $ld(E/A) = ld16 = 4$

- Festellen einer Blockierung (in einem 4x2er Block Fenster stimmen alle 4 Spalten überein)

$E_1 \to A_6$ und $E_{14} \to A_6$ =00010110
11100110 4.Stufe Blockierung am Ausgang A6
$E_1 \to A_6$ und $E_{13} \to A_1$ =00010110
11010001 keine Blockierung
$E_2 \to A_{13}$ und $E_0 \to A_{12}$ =00101101
1100 3. Stufe Blockierung Ausgang A6

6.7.5 Prinzip Paketvermittlung

- Nachricht wird in Pakete geteilt
- Pakete werden unabhängig voneinander über das Netzwerk vom Sender zum Empfänger transportiert
- Paket besteht aus drei Teilen:
  - Header (enthält Routing- und Kontrollinformationen)
  - Datenteil (enthält Anteil der Nutzdaten/Gesamtnachricht)
  - Endstück (Trailer: Fehlerkorrekturcode)
- Drei Verfahren:
  - Store-and-forward Prinzip
    - gesamtes Paket wird zum Empfänger gesendet
    - Jeder Zwischenknoten speichert das gesamte Paket (store), bevor es weitergesendet wird (forward)
    - Vorteile:
      - schnelle Freigabe von Verbindungen
    - Nachteile:
      - großer latenzzeiten, hoher Speicheraufwand
  - Virtual-Cut-Through-Verfahren (Infiniband)
    - Pakete werden hier piplineartig durch das Netzwerk gesendet
    - Prinzip:
      - Flusssteuerung erfolgt nicht auf Paketbasis sondern auf der Basis wesentlich kleinerer pysikalischer Transporteinheiten, phits (physical units), die mann deshalb auch als flits (= phits = float control units) bezeichnet

            - ein auf dem Übertragungspfad liegender Schalter (Knoten oder Router) betrachtet die ersten phits eines ankommenden Pakets (Routinginfos) → Entscheidung zu welchen Knoten das Paket weitergeleitet wird
            - Ist Verbindung frei, dann wird der Header weitergeschickt
            - restliches Paket wird hinterher geleitet → so dass phits des Paketes pipelineartig auf dem Übertragungspfad liegen
            - Freigabe der Verbindung: → wenn alle phits des Paketes (einschließlich Endstück) vollständig übertragen wurden, wird die Verbindung freigegeben
        * Blockierungen:
            - alle phits des Paketes werden im letzten erreichbaren Knoten aufgesammelt (store-and-forward-artig)
        * Vorteile
            - geringere Latenzzeiten, aber man muss viel Speicher vorhalten
    - Worm-Hole Routing
        * unterscheidet sich vom Virtual Cut Through nur in der Behandlung von Blockierungen:
            - Header blockeirt so lange bis Verbindung wieder frei ist
            - alle nachfolgenden flits werden auch blockiert und verbleiben in ihrer Position → flits des Paketes liegen wie ein Wurm in der Leitung
        * Vorteile:
            - niedrige Latenzzeit
            - am wenigsten Speicherbedarf
        * Nachteile:
            - Verbindung blockiert komplett im Blockierungsfall

Leitungsvermittlung

- Leitung wird stationär aufgebaut → Overhead → ist für andere blockiert
- nach Aufbau kann man große Datenmengen drüber senden (gut für lange Nachrichten)
- Parallelrechner fast nicht, weil:
    - vor allem kurze Nachrichten

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: $\pi_{\text{Name,Ort}}(\mathit{Studenten})$
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

7.5 SQL

7.5.1 Einleitung

Bestandteile einer Datenbanksprache

Datenbankanfragesprachen

- Typen von Datenbanksprachen
  - Prozedurale Datenbanksprachen
  - Deskriptive Datenbanksprachen

SQL

Structured Query Language (= SQL)

- Standardisierte Datenbanksprache (durch ISO)
- Erfordernisse einer vollständigen DB-Sprache
  - Möglichkeitkeiten zur Datendefinition, Anfrage und Datenänderung (Einfügen, Löschen, Modifizieren einer Menge von Tupeln)
  - Definition von Sichten, physischen Hilfsmitteln, Integritätsbedingungen
  - Zugriffskontrolle im Sinne des Datenschutzes
  - Möglichkeiten zur Kopplung mit einer Wirtssprache

7.5.2 Datendefinitionssprache

SQL Datentypen

Standardtypen (z.B. in ORACLE)

- CHAR
- NUMBER
- VARCHAR
- DATE
- DECIMAL

Spezielle Typen

- CLOB
- LONG RAW und BLOB
- ROWID
- XML

Auswahl von Datentypen: ABC-Regel

Appriopriate (angemessen)
Brief (kurz)
Complete (vollständig)

Anlegen einer Relation (16ff)

Syntax: CREATE TABLE <Relationen-Name>(<Spaltendefinition>[Spaltendefinition]*)
wobei <Spaltendefinition>::=<Attributname><Typ>[NOT NULL]

Ändern und Löschen einer Relation

Syntax ändern: ALTER TABLE <Relationen-Name> ADD <Attributname> <Typ>
Syntax löschen: DROP TABLE <Relationen-Name>

Erzeugen einer Indexstruktur(19)

Syntax: CREATE [UNIQUE] INDEX <INDEX-Name>ON <Relationen-Name>
(<Attributname> [<Ordnung>][, <Attributname> [<Ordnung>]]*) [CLUSTER]
Syntax Index löschen: DROP INDEX <Index-Name>

Erzeugen einer Datenbanksicht (21)

Syntax: CREATE VIEW <Sicht-Name>[(<Attributname>[, <Attributname>]*)] AS <sub-query>
Views zum Umgang mit komplexen und geschachtelten SQL-Abfragen

7.5.3 Datenbankanfragesprache (DQL 25 ff)

7.5.4 Algebra-Operationen in SQL (32 ff)

7.5.5 Sprachelemente jenseits der relationalen Algebra

GROUP BY

- Auswahl benötigter Spalten (Projektion)
- Formelberechnung
- Sortierung
- Gruppierung
- Aggregation

Verfügbare Aggregatfunktionen

- SUM(x)
- AVG(x)
- MAX(x)

- MIN(x)
- COUNT(x)

Aggregatsfunktionen

Transformation, Verdichtung einzelner Tupel zu einem Gesamtwert

- Funktion COUNT() kann auf eine Menge von Tupeln angewendet werden
- Funktionen SUM(), AVG(), MIN(), MAX() können auf eine Menge von Zahen, die als Spalte einer Relation gegeben ist, anagewandt werden
- Zusätzlich stehen die statistischen Größen STDDEV() und VARIANZ() zur Verfügung

Aggregation

Syntax:

```
SELECT <group_columns>, <aggregate_expressions>
FROM <table> [ JOIN <table> ... ]
[ WHERE <condition> ]
GROUP BY <group_columns>
HAVING <aggregate_condition>
```

Hinweis: Alle Spalten bei SELECT, die nicht in einem Aggregat-Ausdruck (mit SUM(), COUNT(), ...) auftauchen, müssen in der GROUP BY Klausel stehen.

Gruppieren und Ordnen

- GOUP BY
- HAVING
- ORDER BY

Beispiel (44)

7.5.6 Innere und äußere Verbundoperationen

Alternative Schreibweise zur "normalen"Verbundoperation

```
SELECT *
FROM x, y
WHERE x.z = y.z

```

```
5 SELECT *
6 FROM x INNER JOIN y ON x.z = y.z
```

Outer-Joins

Unterschieden werden rechte, linke und vollständige OUTER JOINS

7.5.7 Tabellenausdrücke

In der FROM Klausel kann ein beliebiger SELECT Ausdruck stehen

WITH

Muss mehrmals auf die gleiche geschachtelte Tabelle zugegriffen werden, so kann die Tabellendefinition mit WITH ausgeklammert werden

```
1 WITH TempTab (x,y,z) AS (SELECT x,y,z FROM ...)
2
3 SELECT ...
4 FROM TempTab t1, TempTab t2
5 WHERE ...
```

Alias-Namen für Tabellen

Bennenung von Relationen

- Anfragen sollen auch Relationen mit sich selbst verknüpfen können
- Dafür notwendig die Benennung von Relationen → Alias-Namen

Beispiel

```
1 SELECT K1.KName, K1.KAdr
2 FROM Kunde K1, Kunde K2
3 WHERE K1.Kto < K2.Kto
4 AND K2.KName = 'Huber'
```

Guter SQL-Stil

- Einheiten L Einheitenname: l-name
- Bennenung von DB-Objekten
  - Tabelle: Einheiten
  - Index: Einheiten-idx
  - Primärschlüssel ...pk
  - Fremdschlüssel: ...sk

7.5.8 Geschachtelte Anfragen

Prinzip der geschachtelten Anfragen

```
SELECT DISTINCT LName
FROM Lieferant
WHERE Ware IN (SELECT Ware FROM Auftrag WHERE KNAME = 'Huber'
```

Welche Lieferanten liefern alles, was Huber estellt hat?

```
SELECT DISTINCT LName
FROM Lieferant L
WHERE NOT EXISTS
          (SELECT Ware
          FROM Auftrag
          WHERE KName = 'Huber'
                    AND NOT Ware IN (SELECT Ware
                              FROM Lieferant
                              WHERE LName = L.LNAME))
```

Weitere Sprachelemente von SQL (53)

Verwendung von ALL und SOME/ANY Arithmetische Ausdrücke in der SELECT KLAUSEL

7.5.9 Änderungsoperationen Insert, Update, Delete

Einfügen von Tupeln

INSERT: Einfügen von Tupeln Spaltenliste ist optional, wenn alle Spalten angegeben werden

NULL und DEFAULT-Werte

Spalten und Werte müssen nicht angegeben weden, wenn:

- NULL-Werte erlaubt sind oder
- DEFAULT-Werte gestzt sind oder
- AUTO-INCREMENT angegeben wurde

Varianten

Eifügen mehererer Zeilen auf einmal

```
INSERT INTO Studiengaenge (SgNr, Kuerzel, ...)
VALUES
        (1, INF, ...)
        (2, WINF, ...)
```

Löschen und Atualisieren

DELETE: Löschen von Tupeln
UPDATE: Verändern von Tupeln

Beispiel zu Änderungsoperationen 60

```
INSERT INTO

UPDATE
WHERE

DELETE
FROM
WHERE
```

7.5.10 Sequenzen und berechnete Spalten 63 ff

7.5.11 Sicherung der Integrität mit SQL 69ff

DBMS unterstützt die Benutzer bei

- Anfrageverarbeitung
- Persistenter Speicherung der Daten
- Gewährleistung der Konsistenz der Daten

Sicherstellung der ref. Integrität

Regeln beim Ändern/Löschen/Einfügen von Tupeln (74 ff)

Weitere einfache statische Integritätsbedingungen

CHECK CONSTRAINTS (79)
Erzwingen und Missachten von Integritätsbedingungen

7.5.12 Aktive Datenbanktechnologie (80 ff)

ECA-Prinzip eines Triggers

Event-Condition-Action Rules

Anlegen von Triggern (82)

Triggeraktivierungszeit (83 ff)

Datenbankprozeduren und -funktionen (93)

7.6 Entwurfstheorie relationaler Datenbanken

Normalisierung: ER-Modell → Relationen-Schema → Normalisiertes Relationen-Schema mit Hilfe des Synthesealgorithmus

7.6.1 Ziele und Motivation des Datenbankentwurfs

Beispiele 4ff.

7.6.2 Theorie der Funktionalen Abhängigkeiten

Funktionen (16)

Funktionale Abhängigkeiten

Beispiel 17 ff.

Berechnung funktionaler Abhängigkeiten

Aus einer Menge F von FDs sind weitere FDs herleitbar (24ff)

Membership-Problem (26)

7.6.3 Zerlegung (Dekomposition) von Relationen

Zerlegung eines Relationenschemas (29)

Verlustlosigkeit (30 ff.)

Kriterien (32)

Abhängigkeitsbewahrung (33)

Beispiel (34)

7.6.4 Normaleformena

Schemaaqualitätsprüfung

Es gibt Normalformen (NF) die etwas über die Qualität eines Schemas aussagen (37)

Erste NF(39)

Zweite NF(40)

Beispiel zur zweiten NF (40 ff)

Dritte NF(43 ff)

Automatische Zerlegung

Zerlegung wird normalerweise nicht jedes Mal mühsam per Hand gemacht und überführt

Kanonische Überdeckung (46 ff)

Definition: Äquivalenz funktionaler Abhängigketien: Zwei Mengen F und G von FDs eines Relationenschemas R sind äquivalent falls $F^+ = G^+$ gilt

Beispiel Kanonische Überdeckung (49 ff)

3NF-Synthesealgorithmus (52)

Boyce-Codd-Normalform (57)

Erzeugung der BCNF

Mehrwertige Abhängigkeiten (59 ff)

Vierte Normalform (64)

7.7 Transaktionsverarbeitung I - Synchronisation

siehe Skript

7.8 Transaktionsverarbeitung II - Fehlerbehandlung, Logging und Recovery

siehe Skript

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

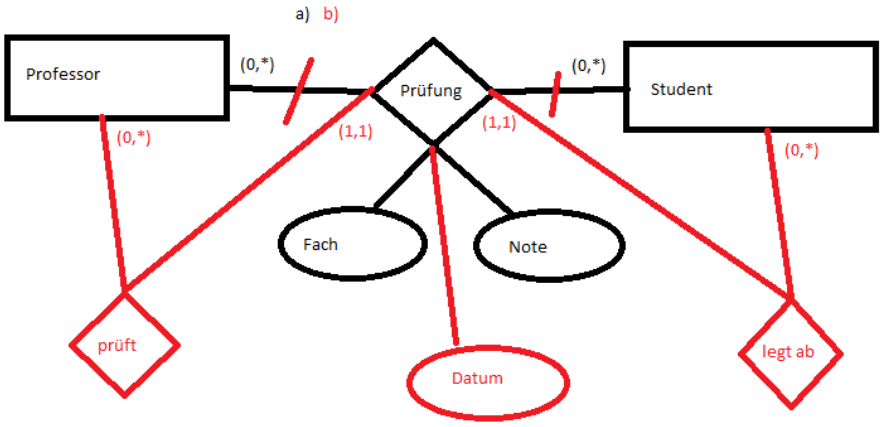


Abbildung 8.1: Professor-Student ER

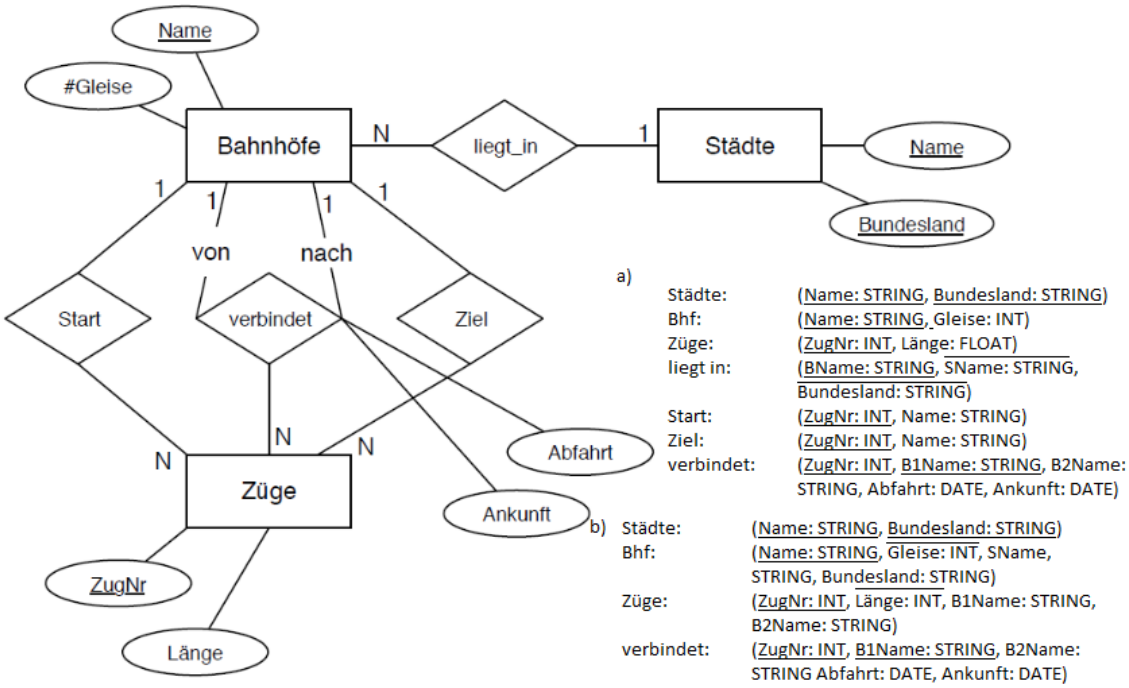


Abbildung 8.2: Bahnnetz ER

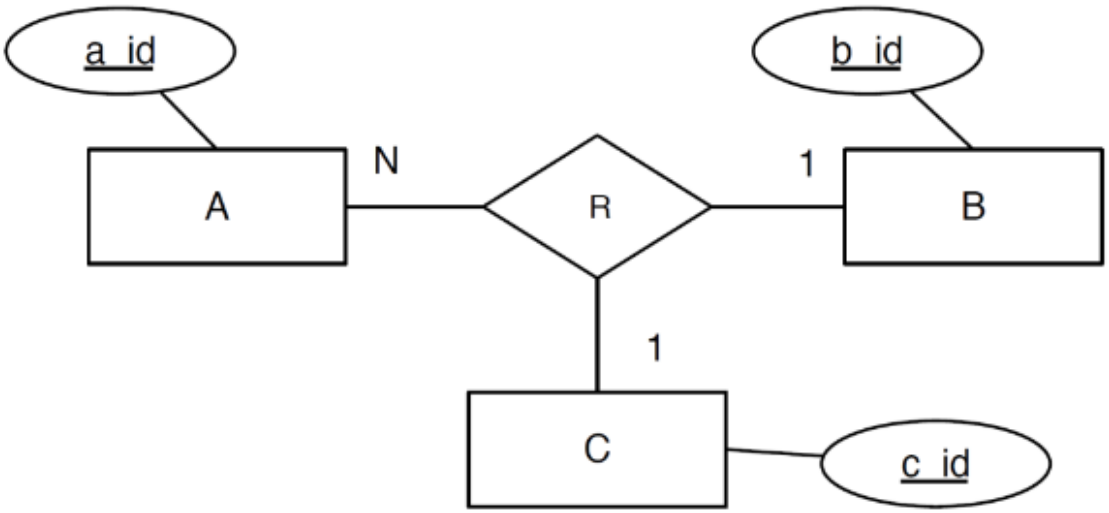


Abbildung 8.3: Relations ER

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

8.3 Relationenalgebra

dennis.koppenhagen@tu-dresden.de
Π → Projektion, Spalte
σ → Selektion, Zeile
ρ → Umbenennung,

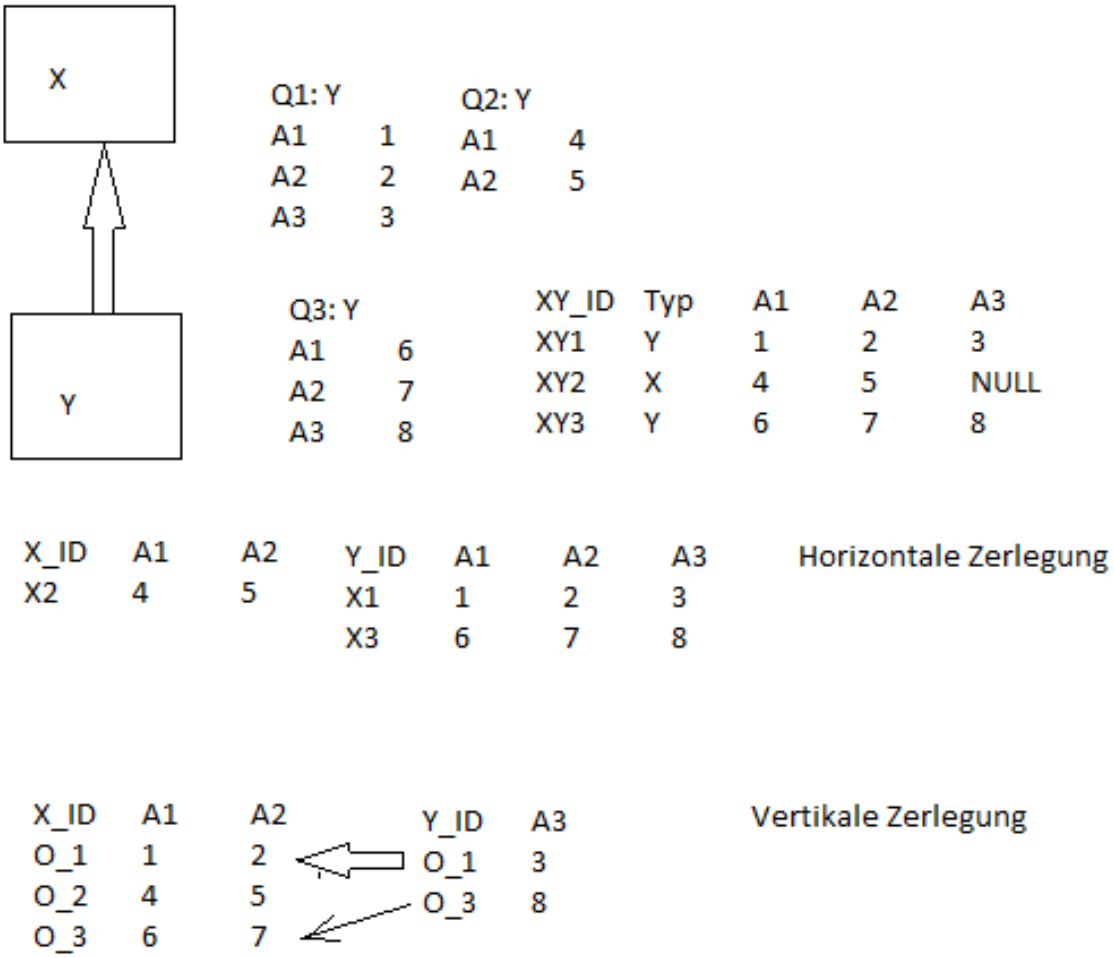


Abbildung 8.4: Relations ER

8.3.1

a) Geben Sie alle Vorlesungen an, die der Student X. gehört hat
$R = \Pi_{\text{Titel, Vorl.Nr.}}(\text{Vorlesung} \bowtie (\text{hören} \bowtie \sigma_{\text{Name = X}}(\text{Studenten})))$

b) Geben Sie die Titel der direkten Voraussetzungen für die Vorlesung Wissenschaftstheorie an:
$R = \Pi_{\text{VVorg.Titel}}(\rho_{\text{VVrg}}(\text{Vorlesung}) \bowtie_{\text{VVorg.VorlNr = Vorgänger}}$
$(\text{Voraussetzen} \bowtie_{\text{VNach.Vorl.Nr = Nachf.}}(\sigma_{\text{VNach.Titel = 'Wiss.Theo'}}(\rho_{\text{VNach}}(\text{Vorlesungen})))))$

c)

$$\Pi_{\text{S1.Name, S2.Name}}\Big(\sigma_{\text{Titel = Grundzüge}}(\text{Vorlesung}) \bowtie_{\text{Vorl.Nr. = h1.Vorl.Nr}}\Big((\rho_{\text{S1}}(\text{Studenten})) \bowtie_{\text{S1.Matr.Nr = h1.Matr.Nr., S1.Matr.Nr != S2.Matr.Nr.}}\Big((\rho_{\text{h1}}(\text{hören})) \bowtie_{\text{h1.Vorl = h2.VorlNr.}}(\rho_{h2}(\text{hören}))\Big) \bowtie_{\text{S2.Matrikel = h2.Matrikel}}(\rho_{S2}(\text{Studenten}))\Big)\Big)$$

8.3.2

a) Finden Sie die Assistenten von Professoren, die den Studenten Fichte unterrichtet haben, z.B. als potentielle Betreuer seiner Diplomarbeit

b) Finden Sie die Studenten, die Vorlesungen hören (bzw. gehört haben), für die ihnen die direkten Voraussetzungen fehlen

8.3.3

a) $R \bowtie S$

A	B	C	D	E	G
1	1	1	1	1	3
2	2	3	2	3	1
2	3	3	2	3	1

b) $R \bowtie\!\subset S$

A	B	C	D	E	G
1	1	1	1	1	3
2	2	3	2	3	1
2	3	3	2	3	1
NULL	NULL	1	3	2	2

c) R $\ltimes$ S

A	B	C	D	E	G
1	1	1	1	1	3
2	2	3	2	3	1
2	3	3	2	3	1
NULL	NULL	1	3	2	2
3	2	2	3	NULL	NULL

d) R $\rtimes$ S

A	B	C	D
1	1	1	1
2	2	3	2
2	3	3	2

8.3.4 Kreuzprodukt und Divisionsoperator

$\Pi_A(R) = \{X, Y, Z\}$
$\Pi_A(R)\text{x}S = \{(X,2),(X,3),(Y,2),(y,3),(Z,2),(Z,3)\}$
$\Pi_A(R)x\S) - R = \{(X,3)\}$
$\Pi_A((\Pi_A(R)xS) - R) = \{X\}$
$\Pi_A(R) - \Pi_A(\ldots) = \{Y, Z\}$

8.4 SQL

8.4.1 Befehle in SQL

1

```
SELECT name, population
FROM city
ORDER BY population DESC
```

2

```
SELECT city
FROM located
WHERE river IS NOT NULL
AND lake IS NOT NULL
```

3

```
SELECT name, population
FROM city
WHERE country = 'D'
ORDER BY population DESC
```

4

```
SELECT DISTINCT l.name
FROM language l
INNER JOIN encompasses e ON l.country = e.country
WHERE e.continent = 'Europe'
```

5

```
fill
```

8

```
SELECT DISTINCT c.name
FROM country c
INNER JOIN city s ON s.country = c.code
INNER JOIN city hs ON c.capital = hs.name
WHERE s.population > hs.population
```

9

```
SELECT c.name, SUM(s.population)
FROM city s
INNER JOIN country c ON s.country = c.code
WHERE s.population > 1000000
GROUP BY c.name
ORDER BY 2 DESC
```

8.5

fill 16.05.

8.5.1 Primzeugs

a) Prim 1

```
SELECT p, p+2, p+4
FROM Prim
WHERE p + 2 in (SELECT p from PRIM)
          AND p + 4 in (SELECT p FROM PRIM)
```

b) Prim 2

```
SELECT p
FROM Prim
WHERE ((p+1) mod 6 = 0 or (p-1) mod 6 = 0)
          AND p > 3
```

8.6

8.6.1 ER-Modell

MIN/MAX Angaben 8.5

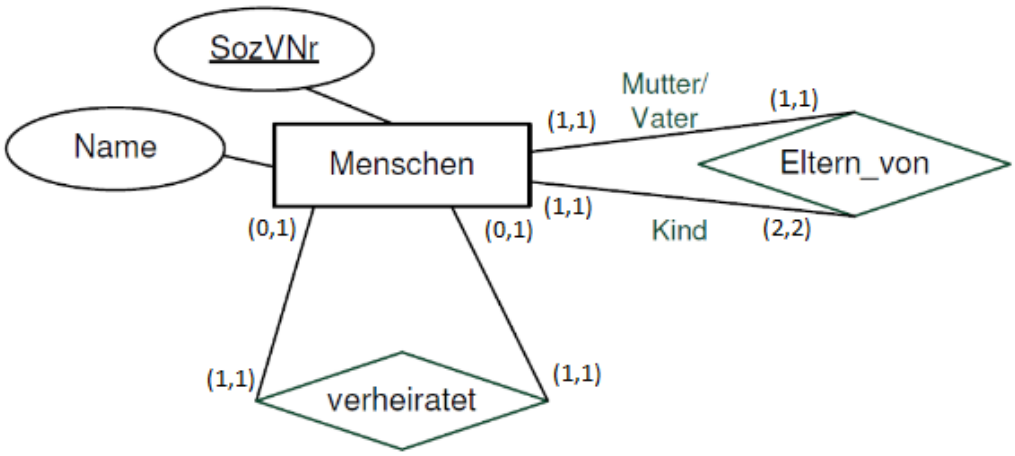


Abbildung 8.5: Min/Max

```
CREATE TABLE Menschen (SozVNr varchar(30) primary key,
          Name varchar(30));
```

verheiratet

```
CREATE TABLE verheiratet(
          Ehepartner1 varchar(30)
          REFERENCES Menschen ON DELETE CASCADE,
          Eheparnter 2 varchar(30) NOT NULL UNIQUE
          REFERENCES Menschen ON DELETE CASCADE, PRIMARY KEY(Ehepartner1)
);
```

Eltern von

```
CREATE TABLE Eltern_von (
        MutterVater varchar(30) NOT NULL REFERNECES Menschen,
        Kind varchar(30) REFERNECES Menschen NOT NULL,
        PRIMARY KEY(VaterMutter, Kind));
```

8.6.2 SQL-Anfragen

a) AB ist Schlüssel? Wenn ja dann darf folgendes keine Einträge haben.

```
SELECT A,B FROM R GROUP BY A, B HAVING COUNT(*) > 1;
```

b) DE $\rightarrow$ B

```
SELECT D, E FROM R GROPU BY D, E HAVING COUNT (DISTINCT B) > 1;
```

8.6.3 Relationsschema

a)
- {Name, Aufbage} $\rightarrow$ {Erzielt}
- {Name } $\rightarrow$ {KlausurSumme, Bonus}
- {KNote, Bonus} $\rightarrow$ {Gnote}
- {KlausurSumme} $\rightarrow$ {KNote}
- {Aufgabe} $\rightarrow$ {Max}

8.7 06.06.2016

8.7.1

Attributhülle: $(F, A) = A^+$
Menge aller Attribute, die funktional von A bestimmt werden.

Links-Reduktion

$\forall \alpha \rightarrow \beta \in F$: Wenn $\beta \subseteq$ Attributhülle $(F, (\alpha - A))$ dann entferne A auf linker Seite, $A \in \alpha$

Rechts-Reduktion

$\forall \alpha \rightarrow \beta \in F$: Wenn B $\in$ Attributhülle $(F \rightarrow (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta \rightarrow$ B)) $\cup \alpha)$ dann entferne B auf rechter Seite, $B \in \beta$

Aufgabe

Beispiel Links- und Rechtsreduktion

$$F = \{A \to B, B \to C, AB \to C\}$$
nach Linksreduktion: $F = \{A \to B, B \to C, A \to C\}$
nach Rechtsreduktion: $F = \{A \to B, B \to C\ A \to \emptyset\}$
$= \{A \to B, B \to C\}$ wird kanonische Überdeckung genannt

a) $\emptyset \to \{MaxSumme\}$

b) FDS:

- {KNote, Bonus} → {GNote}
- {Aufgabe} → {Max}
- {ErzieltSumme} → {KNote}
- {Name, Aufgabe} → {Erzielt}
- {Name} → {ErzieltSumme, Bonus, GNote}
  nach Rechtsreduktion: {Name} → {ErzieltSumme, Bonus}
- Formal funktioniert Reduktion so:
  {Name} → {ErzieltSum, Bonus, GNote}
  {ErzielstSum} ∉ AttributHülle{FD, F - {Name} → {ErzieltSum, Bonus, GNote} ∪ ({Name}→{Bonus,GNote}),{Name}),

c)
- Noten: {[KNote,Bonus, GNote]}
- Aufgaben: {[Aufgabe, Max]}
- PunkteNote: {[ES, KNote]}
- MaxSumme: {[MaxSumme]}

8.7.2

- {Signatur} → {Titel}
- {Vorgang} → {Datum}
- {Datum,Vorgang} →{Benutzer}
- {Benutzer} → {Straße, PLZ, Ort}
- {PLZ} → {Ort}

PK: {Vorgang, Signatur}

- R1: {Vorgang, Datum, Benutzer, Straße, PLZ, Ort}
- R2: {Signatur, Titel}
- R3: {Vorgang, Signatur}

ist in 2. NF

8.7.3 FLüge

FLÜGE(E,G,A0,Z0,AZ,D)

ReiseBüros(K,N,O,A)

Passagiere(V,N,W)

Reserviert(K, F)

BuchtBei(V,N, K)

FliegtMit(V, N, F)

$F^+ = F \cup G \cup A0 \cup Z0 \cup AZ \cup D$

$G \cup A0 \cup Z0)^+ = G \cup A0 \cup Z0 \cup F \cup AZ \cup D$

$G \cup A0 \cup Z0 \rightarrow F$

8.8

8.8.1

A tomicity

C onsistency

I solation

D urability

8.8.2

DIRTY READ

T2	T3
	B0T3
	r2(B)
	w3(B)
B0T2	
r2(B)	
r2(A)	
w2(B)	
C2	
	a3

NON REPEATABLE READ

T1	T2
B0T1	
r1(B)	
	B0T2
	r2(B)
	r2(A)
	w2(B)
	C2
w1(A)	
r1(B)	
w1(B)	
C1	

8.8.3

a) ...

b) Serialisierbarkeitsgraph

- T1
  - T1 → T2
  - T1 → T3
- T2
  - T2 → T1
  - T2 → T3
- T3

c) ist nicht serialisierbar ($\exists$ Zyklen)

8.8.4

a) w1[x]w1[y]c1r2[x]w2[z]c2r3[y]a3

(1) SR (keine Zyklen)?

- T1
  - T1 $\rightarrow$ T2
  - T1 $\rightarrow$ T3
- T2
- T3

$\Rightarrow$ der Graph hat keine Zyklen

(2) RC: schreib-lese Reihenfolge = schließ Reihenfolge

ja $\rightarrow$ ACA: Lesen nur von abgeschlossenen Transaktionen

ja $\rightarrow$ ST: Schreiben nur auf abgeschlossenen Transaktionen

b)

c)

- T1
  - T1 $\rightarrow$ T2
  - T1 $\rightarrow$ T3
- T2
  - T2 $\rightarrow$ T1 $\Rightarrow$ ZYKLUS!!! Schreib-Lese-Reihenfolge?
    - nicht RC $\rightarrow$ nicht ACA und nicht ST
- T3

8.8.5

a) $w_1(x)r_2(x)w_2(y)c_2c_1$ Historie nicht rücksetzbar aber korrekt bei 2PL $\rightarrow$ deshalb S2PL, damit wird der Term serialisierbar

b) S2PL erzeugt serielle Abläufe, dagegen 2PL serialisierbar aber **ohne** serielle Abläufe zu garantieren!!

T1	T2
B0T1 Lock X(A) w1	- - -
- - -	B0T2 Lock X(B) W2(B)
Lock S(B) -	- Lock S(A)

8.8.6

	a	b	c	d	e	f
T1	x	s	**X**	-	s	-
T2	-	s	-	-	-	**S**
T3	-	**X**	s	-	s	-
T4	**S**	-	s	x	s	-
T5	-	-	-	**X**	-	x

- T1
  - T1 → T3
  - T1 → T4
- T2
  - T2 → T5
- T3
  - T3 → T1
  - T3 → T2
- T4
  - T4 → T1
- T5
  - T5 → T4

Teil V

Hardware Laboratory

Teil VI

C++4CG

Literaturverzeichnis