Theoretische Informatik

github/bircni

Inhaltsverzeichnis

1 Mathematische Grundbegriffe

1.1 Mengen

- Eigenschaften von Mengen
  Eine Menge ist eine Sammlung von Elementen, diese können alles mögliche sein.
  Wesentliche Eigenschaften von Mengen:
  - Mengen haben keine Anordnungsreihenfolge
  - Ein Element kann höchstens einmal in einer Menge enthalten sein
    Bsp. $a, a, b = a, b$
- Grundmengen von Zahlen
  - $\mathbb{N}$: Natürliche Zahlen
  - $\mathbb{Z}$: Ganze Zahlen
  - $\mathbb{Q}$: Rationale Zahlen
  - $\mathbb{R}$: Reelle Zahlen
- Mengenoperationen
  - $\emptyset$: leere Menge
  - $x \in A$: x ist Element der Menge A
  - $|A|$: Kardinalität der Menge, bzw. Anzahl der Elemente
  - $A \subseteq B$: A ist Teilmenge von B oder gleich der Menge von B
  - $A \subset B$: Menge A ist echt in Menge B enthalten
  - $A \cup B$: **Vereinigung:** Menge aller Elemente in A oder B
  - $A \cap B$: **Schnittmenge:** Menge aller Elemente, sowohl in A als auch in B
  - $A \backslash B$: **Differenzmenge:** alle Elemente, die in A aber nicht in B enthalten sind
  - $A \times B$: **kartesisches Produkt:** Menge aller Paare, die aus A und B gebildet werden können
  - $\mathcal{A}$: **Potenzmenge:** Menge aller Teilmengen von A
- Kardinalität der Potenzmenge
  Für die Potenzmenge $\mathcal{M}$ einer endlichen Menge M gilt: $|\mathcal{M}| = 2^{|M|}$

1.2 Relationen

Eine Relation ist eine Beziehung zwischen Elementen einer Menge.
Eine zweistellige Relation R über einer Grundmenge M ist eine Menge von Paaren (x,y) mit $x \in M$ und $y \in M$, d.h. $R \subseteq M \times M$

1.3 Funktionen

- Eine Funktion $f : A \rightarrow B$ ist eine zweistellige Relation $f \subseteq A \times B$ mit der Eigenschaft:
  $(a, b_1) \in f(a, b_2) \in f \Rightarrow b_1 = b_2$
- Ist $(a, b) \in f$, dann heißt b Funktionswert zu a, $f(a) = b$
- Eine Funktion $f : A \rightarrow B$ heißt **total**, wenn es für jedes Argument $a \in A$ einen Funktionswert $f(a) = b \in B$ gibt. Sonst heißt die Funktion **partiell**.
- Gibt es für ein $a \in A$ keinen Funktionswert , dann ist f an dieser Stelle **undefiniert** $f(a) = \perp$

1.4 Unendliche Mengen

- **abzählbar unendlich**: Gleich mächtig, wie die Menge der natürlichen Zahlen.
  Also: für jedes Element gibt es eine Position und für jede Position i gibt es auch einen Wert $m_i \in M$
- **überabzählbar unendlich**: unendlich und nicht abzählbare Menge
  Keine 1-zu-1-Zuordnung zu den natürlichen Zahlen
  Bsp: Die Menge der unendlich langen 0/1 Folgen, die Potenzmenge der natürlichen Zahlen

2 Grundbegriffe der Graphentheorie

2.1 Gerichtete Graphen

2.1.1 Definition

- Ein gerichteter Graph $G = (V, E)$ besteht aus
  V Menge von Knoten
  $E \subseteq V \times V$ Menge von Kanten
- Für eine Kante $e = (u, v)$ ist u der Ausgangs- und v der Zielknoten
- Existitert eine Kante $e = (u, v)$, dann ist v ein Nachbar von u. Sie sind adjazent.
- Eine Kante mit gleichem Ausgangs- und Zielknoten heißt Schlinge/Schleife.

2.1.2 Diagrammdarstellung von Graphen

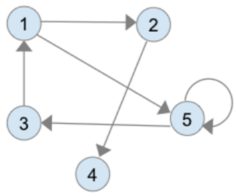
- Knoten werden als Kreise dargestellt.
- Kanten werden als Pfeil vom Ausgangs- zum Zielknoten dargestellt.
- Die Positionierung von Knoten ist irrelevant, Kanten müssen keine geraden Linien sein.

2.1.3 Weitere Darstellungsmöglichkeiten von Graphen

Graphen können als Adjazenzmatrix oder in Adjazenzlisten dargestellt werden:

	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	1	0
3	1	0	0	0	0
4	0	0	0	0	0
5	0	0	1	0	1

Knoten	*Adjazenzliste*
1	[2, 5]
2	[4]
3	[1]
4	[]
5	[3, 5]



2.1.4 Knotengrad

Für einen Knoten eines gerichteten Graphen $G = (V, E)$ ist

- der **Eingangsgrad** die Anzahl der Zielknoten v
- der **Ausgangsgrad** die Anzahl der Ausgangsknoten v
- der **Grad** die Summe von Ausgangsgrad und Eingangsgrad von v

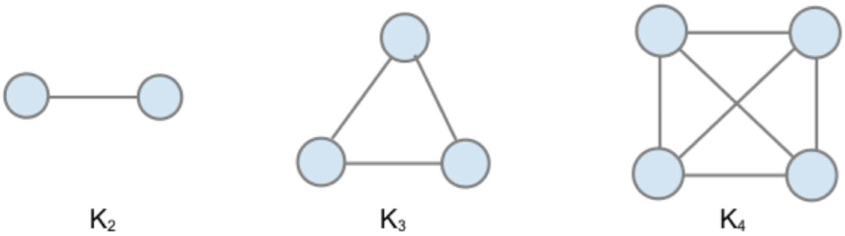
2.2 Ungerichtete Graphen

2.2.1 Definition

- Ein ungerichteter Graph $G = (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge E
- Der Grad eines Knotens ist die Anzahl der von v ausgehenden Kanten. (Schlingen werden doppelt gezählt)

2.2.2 Vollständige Graphen

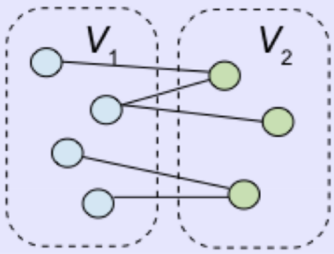
Ein ungerichteter Graph heißt **vollständiger Graph**, wenn es zw. je zwei verschiedenen Knoten eine Kante gibt.



Ein vollständiger Graph mit **n** Knoten hat $\frac{n(n-1)}{2}$ Kanten.

2.2.3 Bipartite Graphen

- Ein Graph $G = (V, E)$ heißt **bipartit**, wenn die Knotenmenge V in zwei disjunkte Teilmengen V_1, V_2 mit $V = V_1 \cup V_2$ aufgeteilt werden kann, so dass jede Kante zwei Knoten aus verschiedenen Teilmengen verbindet.



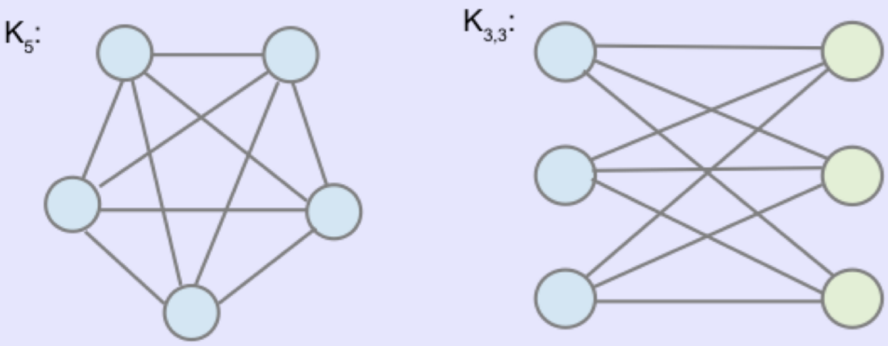
- Ein **bipartiter Graph** heißt **vollständig**, wenn es von jedem Knoten aus V_1 eine Kante zu jedem Knoten aus V_2 gibt.

2.2.4 Planare Graphen

Definition:
Ein Graph G heißt **planar**, wenn er in einer Ebene so gezeichnet werden kann, dass sich keine Kanten kreuzen.

Kuratowski-Graphen

(1) Der **vollständige Graph K_5** mit 5 Knoten und der **vollständige bipartite Graph $K_{3,3}$** mit zweimal drei Knoten sind **nicht planar**.



(2) Ein endlicher Graph ist genau dann planar, wenn er keinen Teilgraphen erhält, der durch Unterteilung von K_5 oder $K_{3,3}$ entstanden ist. Unterteilung bedeutet hier, dass beliebig oft (auch null mal) neue Knoten auf einer Kante eingefügt werden.

- Wenn in einem Graphen also irgendwie K_5 oder $K_{3,3}$ als Teil einhalten ist, ist der Graph nicht planar.
- Folglich sind also beispielsweise alle vollständigen Graphen mit mehr als 5 Knoten auch nicht planar, da immer K_5 als Teilgraph enthalten ist.

2.3 Wege, Zyklen und Kreise

2.3.1 Definition

- Ein **Zyklus** ist ein Weg der Länge n¿0 - Ausgangs und Endknoten stimmen überein
- Ein **Kreis** ist ein Zyklus, bei dem kein Knoten doppelt besucht wird
- Ein gerichteter Graph heißt **azyklischer Graph**, wenn er keinen Zyklus enthält

2.4 Zusammenhang

2.4.1 Zusammenhang bei ungerichteten Graphen

Ein ungerichteter Graph heißt **zusammenhängend**, wenn es von jedem Knoten zu jedem anderen mind. einen Weg gibt.
Ein Teilgraph des ungerichteten Graphen heißt **Zusammenhangskomponente**, wenn folgende Bedingungen gelten:

- G' ist zusammenhängend
- G hat keinen größeren Teilgraphen G", der zusammenhängend ist und G' als Teilgraph enthält

Die Knotenmenge von G' ist eine Teilmenge der Knotenmenge von G

2.4.2 Zusammenhang bei gerichteten Graphen

stark zusammenhängend: für jede Kombination von v_1, v_2 gibt es jeweils einen Weg
(schwach) zusammenhängend: wenn er als ungerichteter Graph zusammenhängend wäre

2.5 Euler-Zyklen und -Wege

2.5.1 Definition

- Ein **Euler-Weg** für G ist ein Weg in dem jede Kante genau einmal durchlaufen wird
- Ein **Euler-Zyklus** für G ist ein Zyklus in dem jede Kante genau einmal durchlaufen wird
- G hat einen **Euler-Zyklus** genau dann, wenn alle Knoten einen **geraden Grad** haben
- G hat einen **Euler-Weg**, wenn ganu zwei Knoten ungeraden Grad haben und alle anderen einen geraden

2.6 Hamilton-Kreise

Ein Zyklus, bei dem jeder Knoten genau einmal besucht wird, nennt man **Hamilton-Kreis**.

2.7 Bäume

2.7.1 Definition

Ein Baum ist ein ungerichterter, zusammenhängender Graph ohne Schlingen.
Alle Knoten mit Grad 1 nennt man **Blätter** des Baums, die anderen heißen **innere Knoten**
Jeder nicht leere Baum mit **n Knoten** hat **n-1 Kanten**

2.7.2 Spannbäume

Ein Spannbaum ist ein Teilgraph eines ungerichteten Graphen, der ein Baum ist und alle Knoten dieses Graphen enthält.
Spannbäume existieren nur in zusammenhängenden Graphen.

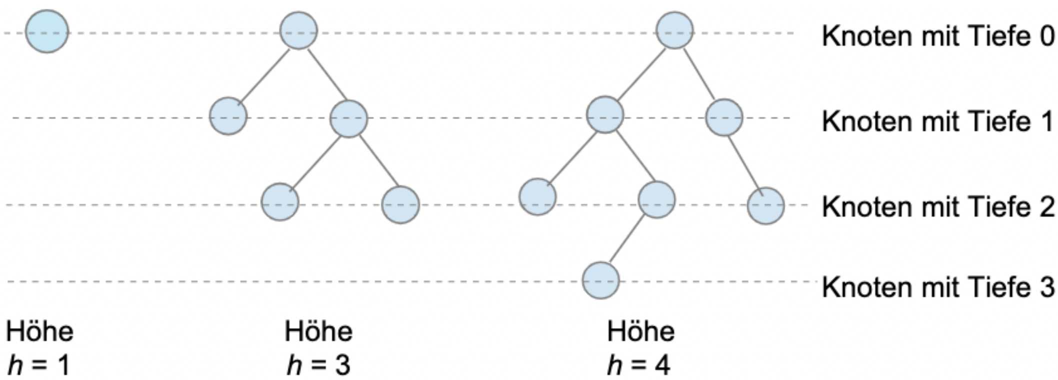
2.7.3 Wurzelbäume

Ein gewurzelter Baum hat einen ausgezeichneten Knoten als **Wurzel**
Jeder Knoten kann beliebig viele Nachfolger haben, diese nennt man Kinder
Jeder Knoten hat genau einen Elternknoten
Blätter sind Knoten ohne Kinder, **innere Knoten** sind Knoten mit Kindern
Definition:
Ein gerichteter, zusammenhängender, azyklischer Graph ist ein Wurzelbaum, wenn

- es genau einen Knoten w mit Eingangsgrad 0 gibt (Wurzel)
- alle anderen Knoten den Eingangsgrad 1 haben

2.7.4 Tiefe von Knoten und Höhe von Bäumen

Die **Tiefe** eines Knoten ist die Länge des Pfades vom Wurzelknoten zum Knoten k
Ein leerer Baum hat die Höhe h=0
Ein Baum aus einem einzigen Knoten hat die Höhe h=1



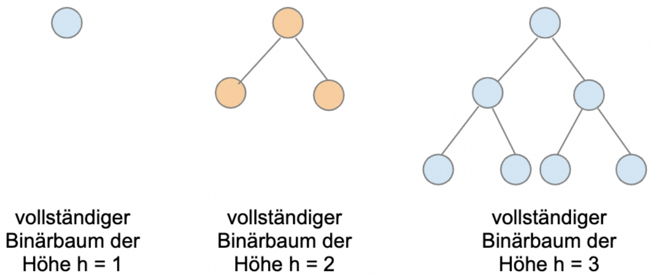
2.8 Binärbäume

Ein Binärbaum ist ein Wurzelbaum, dessen Knoten maximal den Ausgangsgrad 2 haben, d.h. ein Knoten hat maximal 2 Kinder

2.8.1 Vollständige Binärbäume

Ein Binärbaum heißt **vollständig**, wenn

- alle innere Knoten den Ausgangsgrad 2 haben
- alle Blätter die gleiche Tiefe haben



Eigenschaften:
Ein vollständiger Binärbaum der Höhe $h > 0$ hat

- $2^{(h-1)}$ **Blätter**
- $2^h - 1$ **Knoten**

2.8.2 Traversierung von Binärbäumen

- Preorder-Traversierung (W-L-R)
- Inorder-Traversierung (L-W-R)
- Postorder-Traversierung (L-R-W)

3 Formale Sprachen

3.1 Alphabete und Wörter

Ein **Alphabet** $\sum$ ist eine endliche, nicht leere Menge von Zeichen.

3.1.1 Wörter

Definition

- Ein Wort über einem Alphabet ist eine endlich lange Folge von Zeichen aus $\sum$
- $|w|$ bezeichnet die Länge des Worts w
- ϵ bezeichnet das leere Wort mit Länge 0, $|\epsilon| = 0$
- $|w|_a$ bezeichnet die Anzahl der Vorkommen des Zeichens a in w
- $\sum^*$ bezeichnet die Menge aller Wörter, die mit Zeichen von $\sum$ gebildet werden können

3.1.2 Konkatenation

Die Konkatenation (Aneinanderhängen) von zwei Wörter wird als $v \cdot w$ (vw) notiert.

- **Assoziativität**:
  $u \cdot (v \cdot w) = (u \cdot v) \cdot w$
- ϵ ist **neutrales Element**:
  $\epsilon \cdot w = w \cdot \epsilon = w$
- Addition der Längen:
  $|v \cdot w| = |v| + |w|$
- **n-fache Konkatenation**:
  $w^0 = \epsilon$
  $w^n = w \cdot w^{n-1}$ für $n > 0$
  $|w^n| = n \cdot |w|$

3.2 Sprachen

Die **Syntax** einer Sprache beschreibt die Regeln wie "Äußerungen" der Sprache gebildet werden können.
Die **Semantik** beschreibt die Bedeutung der formulierbaren Äußerung.
Die **Pragmatik** beschäftigt sich mit der Nutzung der Sprache.
Eine Sprache L über dem Alphabet $\sum$ ist eine Menge von Wörtern über $\sum$

Einige Sprachen über Alphabet Σ = {a, b}:

- L_1 = {ε, ab, ba} — endliche Sprache, besteht nur aus drei Wörtern.
- L_2 = {b, bb, bbb, bbbb, …} — Sprache beliebig langer Folgen von b, unendliche Sprache
- L_3 = {} — leere Sprache, enthält gar kein Wort
- L_4 = {ε} — Sprache, die nur das leere Wort enthält
- L_5 = {ε, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, baab, bbbb, ...}

3.2.1 Operationen für Sprachen

Definition

- Die **Konkatenation** $L_1 \cdot L_2$ zweier Sprachen L_1 und L_2 ist
  $L_1 \cdot L_2 = vw|v \in L_1, w \in L_2$
- Für eine Sprache L ist L^n wie folgt definiert:
  $L^0 = \epsilon$
  $L^n = L \cdot L^{n-1}$, für $n > 0$
- Die **Kleene'sche Hülle** L^* einer Sprache L ist definiert durchlaufen
  $L^* = L^0 \cup L^1 \cup L^2 \cup ...$

3.3 Exkurs: XML

Serialisierung = Abbildung strukturierter Daten in eiene sequentielle Darstellungsform

- **XHTML**: Beschreibung von Web-Seiten
- **SVG**: zweidimensionale Vektorgrafik
- **ODF**: genormtes Austauschformat für Bürodokumente
- **MathML**: Dokumentenformat zur Darstellung mathematischer Formeln
- **MusicXML**: offenes Dateiformat zum Austausch von Musiknoten

- **WSDL**: Schnittstellen-Beschreibung für Web-Services

3.3.1 Wohlgeformtheit

Ein Text ist ein wohlgeformter XML-Text, wenn er folgende Regeln erfüllt:

- Ein XML-Text besteht aus genau einem XML-Element
- Ein XML-Element beginnt mit einem Anfangstag ¡tag¿ und endet mit dem gleichnamigen Endtag ¡/tag¿
- Elementare Texte können beliebige Zeichenfolgen sein, die aber keine Tags enthalten

3.3.2 Weitere XML-Details

- Anfangstags können mit Attributen versehen werden, um zusätzliche Informationen mit einem Knoten des Baums zu verbinden.

```
<person alter="24">  … </person>
```

- Steht zwischen Anfangstag und Endtag kein Inhalt, kann das Element zu einem Tag mit < ... /> zusammengefasst werden

```
<image name="hochschule.jpg" />
```

entspricht

```
<image name="hochschule.jpg"></image>
```

- Eine XML-Datei beginnt immer mit Angaben zu XML-Version, Zeichencode und ggf. weiteren Metainformationen, z.B. Strukturdefinitionen.

```
<?xml version=1.0" encoding="ISO-8859-1"?>
<!DOCTYPE adressBuch SYSTEM "adressBuch.dtd">
  …
```

Danach folgt dann das Wurzelelement als eigentlicher Inhalt.

4 Reguläre Ausdrücke

4.1 Syntax reguläre Ausdrücke

Die Menge der **regulären Ausdrücke** über einem Alphabet ϵ ist rekursiv so definiert: Folgendes sind elementare reguläre Ausdrücke:

(1) $\emptyset$

(2) ϵ

(3) **a** für jedes Zeichen $a \in \epsilon$

Sind R und S beliebige reguläre Ausdrücke, dann sind auch folgendes reguläre Ausdrücke:

(4) $R \cdot S$

(5) $R|S$

(6) R^*

(7) (R)

Notationskonvention

- Operator * bindet stärker als Operator $\cdot$
- Operator $\cdot$ bindet stärker als Operator $|$

4.2 Semantik reguläre Ausdrücke

Die von einem regulären Ausdruck R dargestellte Sprache L(R) ist folgendermaßen definiert:

(1) $L(\emptyset) =$ // leere Sprache, enthält gar kein Wort

(2) $L(\epsilon) = \epsilon$ // Sprache, die nur das leere Wort enthält

(3) $L(a) = a$ // Sprache, die nur das Wort aus dem einzelnen Zeichen a enthält

(4) Ist $L(R) = L_1$ und $L(S) = L_2$ dann ist $L(R \cdot S) = L_1 \cdot L_2$

(5) Ist $L(R) = L_1$ und $L(S) = L_2$ dann ist $L(R|S) = L_1 \cup L_2$

(6) Ist $L(R) = L_1$ dann ist $L(R^*) = L_1^*$

(7) Ist $L(R) = L_1$ dann ist $L((R)) = L_1$

Die Bedeutung zusammengesetzter Ausdrücke kann so erklärt werden:

- $R|S$ ist die Menge aller Wörter, die in der Sprache R oder in der Sprache S enthalten sind. Sprechweise: "R oder S"
- **RS** ist die Menge aller Wörter, so dass der erste Teil des Worts in der Sprache von R liegt und der Rest in S
- R^* ist die Menge aller Wörter $w = u_1 u_2 ... u_n$ die sich aus bel. vielen Teilwörtern zusammensetzen

R^+ ist die Abkürzung für RR^* $R?$ ist die Abkürzung für $R|\epsilon$

4.3 Äquivalenz regulärer Ausdrücke

Zwei reguläre Ausdrücke R und S heißen äquivalent, falls $L(R) = L(S)$, d.h. wenn beide Ausdrücke die gleiche Sprache darstellen.
Eigenschaften
Für beliebige reguläre Ausdrücke R, S und T gilt:

- $R|S = S|R$
- $(R|S)|T = R|(S|T)$
- $RS|RT = R(S|T)$
- $RT|ST = (R|S)T$
- $R|R = R$
- $R|\emptyset = \emptyset|R = R$
- $(RS)T = R(ST)$
- $R\epsilon = \epsilon R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R^{**} = R^*$
- $R^* = \epsilon|RR^*$ bzw. $R^* = \epsilon|R^+$
- $R^* = \epsilon|R^*$

4.4 Anwendung regulärer Ausdrücke

boolean	**matches (regex)** prüft, ob der String entsprechend dem regulären Ausdruck regex aufgebaut ist.
String	**replaceAll (String regex, String replacement)** ersetzt alle Vorkommen des Musters regex durch replacement
String	**replaceFirst (String regex, String replacement)** ersetzt das erste Vorkommen des Musters regex durch replacement
String []	**split** (String regex, **int limit**) Splits this string around matches of the given regular expression

5 Kontextfreie Grammatiken