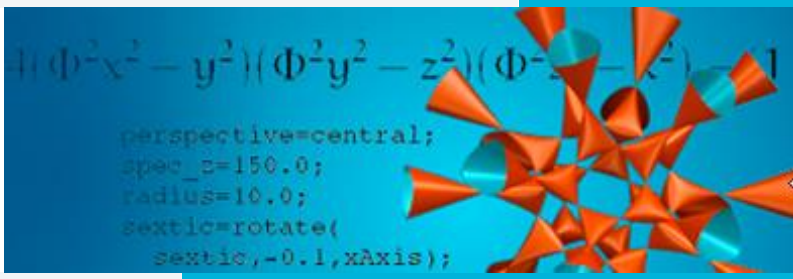
Philipps Universität Marburg

Theoretische Informatik: Alphabete, Worte, Sprachen

Prof. Dr. Elmar Tischhauser



Basierend auf Material von © H.-P. Gumm und J. Daubert

Alphabete, Worte, Sprachen

1. Alphabete und Worte

   – Definitionen, Beispiele
   – Operationen mit Worten
   – Induktionsbeweise

2. Sprachen

   – Definition und Beispiele
   – Operationen auf Sprachen
   – Reguläre Sprachen

3. Spracherkennung

   – Entscheidbarkeit
   – Semientscheidbarkeit

Mathematik-Refresher: Logik

- Aussage: kann **wahr** oder **falsch** sein
- Verknüpfung von Aussagen mit ¬ (NOT), ∧ (AND), ∨ (OR)
- Implikation $A \Rightarrow B$: „wenn A dann B“
  - Äquivalent zu $\neg A \vee B$
  - A hinreichend für B
  - B notwendig für A
  - Äquivalent zu $\neg B \Rightarrow \neg A$
- $A \Leftrightarrow B$: „A genau dann wenn B“
  - Äquivalent zu $(A \Rightarrow B) \wedge (B \Rightarrow A)$

- Prädikat $P(x_1, \ldots, x_n)$: Aussage (wahr/falsch) nach Substitution aller Variablen mit Elementen aus ihren Universen
- Quantoren
  - Allquantor: $\forall x. P(x)$
  - Existenzquantor: $\exists x. P(x)$

Mathematik-Refresher: Mengen

- Menge: unsortierte Sammlung von Elementen
- $x \in M$ falls x Element von M, $x \notin M$ sonst.
- Beschreibung von Mengen
  - durch Aufzählung: $M = \{1,2,3\}$
  - durch Prädikate: $\{x \in \mathbb{U}: P(x)\}$ mit Universum U, z.B. $M = \{x \in \mathbb{N}: 1 \leq x \leq 3\}$
- Leere Menge: $\emptyset$ oder {}
- Kardinalität einer Menge (=Anzahl der Elemente falls endlich):
  - $|M|$ oder $\#M$
  - $|\mathbb{N}|$: abzählbar unendlich
  - $|\mathbb{R}|$: überabzählbar unendlich
- Untermenge $A \subseteq B: \forall x. x \in A \Rightarrow x \in B$
- Gleichheit: $A = B$ falls $\forall x. x \in A \Leftrightarrow x \in B$
  - Äquivalent zu $A \subseteq B \wedge B \subseteq A$
- Potenzmenge $\mathcal{P}(M)$: Menge aller Teilmengen von M

Mathematik-Refresher: Mengen (II)

- Verknüpfungen:
  - Schnitt

$$A \cap B = \{x : x \in A \wedge x \in B\}$$

  - Vereinigung

$$A \cup B = \{x : x \in A \vee x \in B\}$$

  - Differenz:

$$A - B = A \setminus B = \{x : x \in A \wedge x \notin B\}$$

  - Komplement:

$$\bar{A} = \{x : x \in U \wedge x \notin A\}$$ für Universum U

- Kartesisches Produkt:
  - $A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$
  - Die (a,b) sind hier geordnete Paare (**Tupel**)

Alphabete

Ein *Alphabet* Σ ist eine endliche Menge

- Die Elemente von Σ nennt man Zeichen oder Symbole

- Für Alphabete benutzen wir große griechische Buchstaben Σ, Γ, …
- Für die Zeichen des Alphabets benutzen wir a,b,c,…

- Beispiele
  - $\Gamma = \{1\}$ – *das unäre Alphabet*
  - $\Sigma = \{ 0,1 \}$ - *das Binäralphabet*
  - $\Sigma = \{ -, ., \square \}$ – das Morsealphabet (lang, kurz, Pause)
  - $\Gamma = \{0,1,\ldots,9,A,B,C,D,E,F\}$ – *das Hex-Alphabet*
  - $\Sigma = \{A,B,\ldots,Z,a,\ldots,z\}$ – *das lateinische Alphabet*
  - $\Sigma = \{\alpha, \beta, \gamma, \ldots\}$ – das griechische Alphabet

Worte

Ein *Wort über einem Alphabet* Σ ist eine endliche Folge von Zeichen aus Σ.

- Spezialfall:
  - das leere Wort = leere Folge von Zeichen
  - Wir schreiben dafür ε. Manche Autoren verwenden λ
  - Programmiersprachen verwenden z.B.: ""
- Beispiele:
  - Worte über {0,1} : ε, 0, 1, 01, 1001, 10101001, …
  - Worte über {a,…,z} : ε, a, abra, anton, jkjhkjh, …

Σ^* := Menge aller Worte über dem Alphabet Σ

- {1}* = { ε, 1, 11, 111, 1111, …}
- {0,1}* = { ε, 0, 1, 00, 01, 10, 11, 000, 001, … }
- {a,…,z}* = { ε, a, b, …, z, aa, ab, …, ba, bb, bc, …}

Wortdarstellung

- Es gibt zwei Methoden nichtleere Worte anzugeben

1. Durch Angabe der Folge aller Zeichen
   - abra, ojeoje, 01001, 7F3D, CAESAR, 11111, …

2. Als $w = a.v$ wobei
   - a das erste Zeichen ($a \in \Sigma$)
   - v das Restwort ist ($v \in \Sigma^*$)

– Beispiel:
   - *abra = a.bra = a.(b.ra) = a.(b.(r.a)) = a.(b.(r.(a.ε)))*

– Auf Klammern kann man verzichten (Warum?)
   - *abra = a.bra = a.b.ra = a.b.r.a = a.b.r.a.ε*

Jedes nichtleere Wort $w \in \Sigma^*$ lässt sich eindeutig darstellen als $w = a.v$ mit $a \in \Sigma$ und $v \in \Sigma^*$.

Operationen auf Worten

- Länge (Anzahl der Zeichen)
  - |w| = Anzahl der Zeichen in w
  - Formale Definition:
    1. $|\varepsilon| = 0$ // Fall: u leer
    2. $|a.v| = 1+|v|$ // Fall: u = a.v

- Konkatenieren (= Aneinanderhängen)
  - $u \circ v = uv$
  - Formale Definition
    1. $\varepsilon \circ v = v$ // Fall: u leer
    2. $(a.u) \circ v = a.(u \circ v)$ // Fall: u = a.v

  - Das Zeichen $\circ$ lassen wir später meist weg.
    - In Java/Scala benutzt man „+“

  - Wir identifizieren das Zeichen a mit dem Wort $a.\varepsilon$ (der Länge 1)
    - Programmiersprachen sind pedantischer : (Char) ´a´ ≠ (String) “a“

  - Für $a \in \Sigma$ und $u \in \Sigma^*$ also: au statt $(a.\varepsilon) \circ u = a.u$
  - Analog ua statt $u \circ a.\varepsilon$

Reverse und Palindrome

- Reverse
  - w^R ist das reverse Wort zu w
  - Formale Definition:

1. $\varepsilon^R = \varepsilon$ // Fall: w=ε
2. $(a.v)^R = v^R \circ (a.\varepsilon)$ // Fall: w=a.v

- Palindrom
  - Wort u mit $u^R = u$
  - Formale Definition:

1. ε ist Palindrom
2. Falls $u \neq \varepsilon$
   1. $a.\varepsilon$ ist Palindrom
   2. $a.v$ ist Palindrom $\Leftrightarrow$ $v = w \circ (a.\varepsilon)$ und w ist Palindrom

Teilworte, Präfixe, Suffixe

- u ist Präfix von $w :\Leftrightarrow \exists\, v\colon\ u \circ v = w$.
  ε präfix $w \Leftrightarrow$ true
  $a.u$ präfix $w \Leftrightarrow w = a.v \wedge u$ präfix v

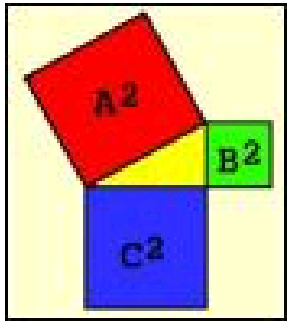
- u ist Suffix von $w :\Leftrightarrow \exists\, v\colon\ v \circ u = w$.
  u suffix $w \Leftrightarrow u^R$ präfix w^R

- t ist Teilwort von $w :\Leftrightarrow \exists\, u, v\colon\ w = u \circ t \circ v$
  t teilwort $w \Leftrightarrow t$ präfix $w \vee w = a.v \wedge t$ teilwort v

Beispiel: Das Wort „kakao“ hat die

Präfixe: {ε, k, ka, kak, kaka, kakao}
Suffixe: {ε, o, ao, kao, akao, kakao}
Teilworte: {ε, k, a, o, ka, ak, ao, kak, aka, kao, kaka, akao, kakao}

Theoreme



- Es gelten u.a. die Theoreme:

1. $\forall u, v, w \in \Sigma^*: (u \circ v) \circ w = u \circ (v \circ w)$ //assoziativ
2. $\forall u, v \in \Sigma^*: |u \circ v| = |u| + |v|$
3. $\forall w \in \Sigma^*: |w^R| = |w|$
4. $\forall u, v \in \Sigma^*: (u \circ v)^R = v^R \circ u^R$

- Wie könnte man so etwas formal beweisen?

Menge aller Worte über Σ

- Σ^* ist kleinste Menge mit
  - $\varepsilon \in \Sigma^*$
  - $\forall a \in \Sigma: \forall u \in \Sigma^*: \quad a.u \in \Sigma^*$

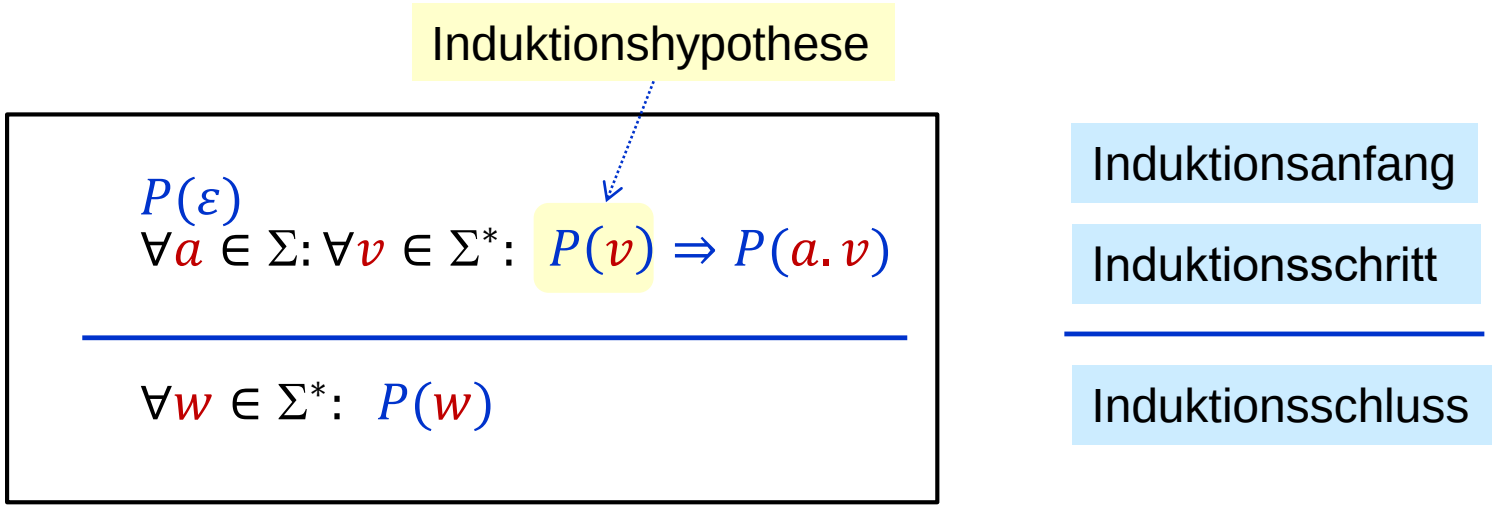
- Gleichheit.
- $\forall u, v \in \Sigma^*: u = v :\Leftrightarrow u = \varepsilon = v$ oder
  $u = a.u'$ und $v = a.v'$ und $u' = v'$

Jedes nichtleere Wort $w \in \Sigma^*$ lässt sich eindeutig darstellen als $w = a.v$ mit $a \in \Sigma$ und $v \in \Sigma^*$.

Induktionsprinzip

- Aus der induktiven Definition von Σ^*:
  - $\varepsilon \in \Sigma^*$
  - $\forall a \in \Sigma,\ \ \forall v \in \Sigma^*: a.v \in \Sigma^*$

- ergibt sich das Induktionsprinzip für Worte:
  - Sei $P: \Sigma^* \rightarrow$ boolean eine „Wort-Eigenschaft“ (**Prädikat**).



Ein Induktionsbeweis

Def von ∘:
$\varepsilon \circ v = v$
$(a.u) \circ v = a.(u \circ v)$

- Behauptung: $\forall w \in \Sigma^*: w \circ \varepsilon = w$
- Wähle: $P(w) := w \circ \varepsilon = w.$

  – Induktionsanfang: P(ε)
  $P(\varepsilon),\ \varepsilon \circ \varepsilon = \varepsilon, \Leftrightarrow true$ // Fall 1 in Def von ∘

  – Induktionsschritt:
  $\forall a \in \Sigma: \forall v \in \Sigma^*: P(v) \Rightarrow P(a.v)$:

  Für beliebige $a \in \Sigma, v \in \Sigma^*$
  sei $P(v)$, d.h. $v \circ \varepsilon = v$ // Induktionshypothese
  Es folgt $(a.v) \circ \varepsilon = a.(v \circ \varepsilon)$ // Fall 2 in Def. von ∘
  $= a.v$ // Induktionshypothese
  Also ist $P(a.v)$ bewiesen
  Also ist $\forall a \in \Sigma: \forall v \in \Sigma^*: P(v) \Rightarrow P(a.v)$ bewiesen

  – Induktionsschluss:
  $\forall w \in \Sigma^*: P(w).$

Wir nehmen an, dass $P(v)$ gilt. Bisher aber nur für $v = \varepsilon$ gezeigt.

Da diese Implikation gilt, können wir nun beliebige Worte $v \in \Sigma^*$ ausgehend von ∅ erzeugen

Aufgaben

- Beweisen Sie die folgenden Behauptungen nach dem Schema der Folie „Ein Induktionsbeweis“.
- Begründen Sie jeden einzelnen Schritt
  - entweder durch Verweis auf einen Fall einer Definition
  - oder durch Verweis auf ein vorher bewiesenes Lemma.

1. $\forall u, v \in \Sigma^*: |u \circ v| = |u| + |v|$
   Hinweis: Induktion über u, d.h.

   $$P(u) \coloneqq \forall v \in \Sigma^*: |u \circ v| = |u| + |v|$$

2. $\forall u, v, w \in \Sigma^*: (u \circ v) \circ w = u \circ (v \circ w)$
   Hinweis: Induktion über u
   Hinweis: $P(u) \coloneqq \forall v, w \in \Sigma^*: (u \circ v) \circ w = u \circ (v \circ w)$

3. $\forall w \in \Sigma^*: |w^R| = |w|$

4. $\forall u, v \in \Sigma^*: (u \circ v)^R = v^R \circ u^R$

Aufgaben

- Zu $v \in \sum^*$ und $a \in \sum$ sei v_+a das Wort, das aus v entsteht, wenn man das Zeichen a hinten anhängt.
  - Geben Sie eine rekursive Definition für die Operation "$_+$" an
  - Zeigen Sie jeweils mit Induktion, dass

1. $\forall a \in \sum: \forall v \in \sum^*: \; v_+a = v \circ a.\varepsilon$
2. $\forall a, b \in \sum: \forall v \in \sum^*: \; (a.v)_+ b = a.(v_+b)$
3. $\forall a \in \sum: \forall u, v \in \sum^*: \; (u \circ v)_+ a = u \circ (v_+a)$

Alphabete, Worte, Sprachen

1. Alphabete und Worte

   – Definitionen, Beispiele
   – Operationen mit Worten
   – Induktionsbeweise

2. Sprachen

   – Definition und Beispiele
   – Operationen auf Sprachen
   – Reguläre Sprachen

3. Spracherkennung

   – Entscheidbarkeit
   – Semientscheidbarkeit

Formale Sprachen

Eine (formale) Sprache über einem Alphabet Σ ist eine Menge von Worten aus Σ^*.

- Jede Teilmenge $L \subseteq \Sigma^*$ ist eine (formale) Sprache
- Insbesondere auch:
  - Σ^* - die Menge aller Worte
  - $\emptyset = \{\}$ - die leere Menge
  - $\{\varepsilon\}$ - die Menge, die das leere Wort enthält
  - $\{a\}$ für $a \in \Sigma$ - ein-elementige Menge mit Wort der Länge 1

Beispiele formaler Sprachen

- Über dem Alphabet { a, b, c }:
  - $L_1 = \{ abba, cbc, a, \varepsilon \}$ - eine Sprache mit 4 Worten
  - $L_2 = \{ a^n b^n \mid n \in \mathbb{N} \}$ - beliebig viele a-s dann gleich viele b-s
  - $L_3 = \{ ucv \mid u, v \in \Sigma^* \}$ - alle Worte, die ein c enthalten
- über dem Alphabet {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}:
  - $L_4 = \Sigma^* - \{\varepsilon\} - \{ 0w \mid w \in \Sigma^* \} \cup \{ 0 \}$ - alle nat. Zahlen in Dezimaldarstellung
- über dem Alphabet { (,) }:
  - L_5 = Menge aller "wohlgeformten" Klammerausdrücke (Dyck-Sprache)

    z.B: $(()(())) \in L_5$ aber $(()))(() \notin L_5$

    wie können Sie diese Sprache präzise spezifizieren ?

„Reguläre“ Operationen auf Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^*$ Sprachen.

$L_1 + L_2 := L_1 \cup L_2$

$L_1 \circ L_2 := \{ u \circ v \mid u \in L_1, v \in L_2 \}$

L^n rekursiv definiert durch

$L^0 := \{ \varepsilon \}$

$L^n := L \circ L^{n-1}$

$L^* := \bigcup_{n \geq 0} L^n$

Den Operator $\circ$ repräsentiert man meist durch Konkatenation: $L_1 L_2$ statt $L_1 \circ L_2$.

Einige Identitäten

- L, R, S seien beliebige Sprachen. Es gelten z. B.:
  - $(L + R)\,S = LS + RS$
    - Beweis: $w \in (L+R)S \Rightarrow w = uv$ mit $(u \in L$ oder $u \in R)$ und $v \in S$

      $\Rightarrow w = uv \in LS$ oder $w = uv \in RS$
    - Rückrichtung analog
  - $S(\cup_{n \in N} R^n) = \cup_{n \in N} (SR^n)$
    - Beweis: $w \in S(\cup_{n \in N} R^n) \Rightarrow w = uv$ mit $u \in S$ und $v \in \cup_{n \in N} R^n$

      $\Rightarrow \exists\, n \in N.\ v \in R^n$

      $\Rightarrow \exists\, n \in N.\ uv \in SR^n$

      $\Rightarrow w = uv \in \cup_{n \in N} SR^n$
    - Rückrichtung : Analog
  - $S(RS)^* = (SR)^*S$
    - Beweis: $w \in S(RS)^* \Rightarrow \exists\, n \in N.\ \exists\, u_0,\ldots,u_n \in S\ \exists\, v_1,\ldots,v_n \in R.$ $w = u_0(v_1u_1)\ldots(v_nu_n)$

      $\Rightarrow w = (u_0v_1)(u_1v_2)\ldots(u_{n-1}\,v_n)\,u_n$

      $\Rightarrow w \in (SR)^*S$

Ausdrückbare Operationen

- Die folgenden Operationen sind mit Hilfe der regulären Operatoren ausdrückbar.

$$L^+ := L\,L^*$$
$$L? := L + \{\varepsilon\}$$

- Alle regulären Operatoren sind monoton, d.h.
  - $L_1 \subseteq M_1$ und $L_2 \subseteq M_2$ $\Rightarrow$ $L_1 + L_2 \subseteq M_1 + M_2$, $L_1 \circ L_2 \subseteq M_1 \circ M_2$
  - $L \subseteq M$ $\Rightarrow$ $L^* \subseteq M^*$
- Alle aus regulären Operatoren zusammensetzbaren Operatoren sind monoton
  - z.B. $^+$, ?

Weitere Operationen auf Sprachen

$L, L_1, L_2 \subseteq \Sigma^*$ Sprachen und $e \in \Sigma$

$L_1 \cap L_2$		Schnitt
Σ^*-L		Komplement
L^R	$:= \{ w^R \mid w \in L \}$	Revers
L_e	$:= \{ v \in \Sigma^* \mid e.v \in L \}$	Ableitung nach e

- Diese Operationen sind **nicht** mit Hilfe der regulären Operatoren ausdrückbar
  - Beweis für Komplement: $L_1 \subset L_2 \Rightarrow \Sigma^*-L_1 \not\subset \Sigma^*- L_2$
  - Andere Op.: Ohne Beweis

Alphabete, Worte, Sprachen

1. Alphabete und Worte

   – Definitionen, Beispiele
   – Operationen mit Worten
   – Induktionsbeweise

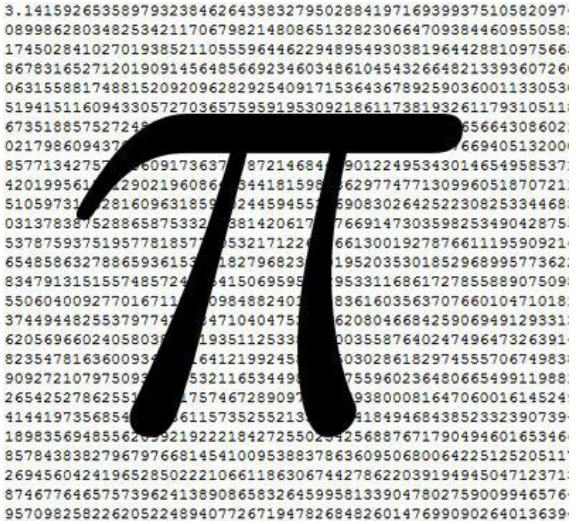
2. Sprachen

   – Definition und Beispiele
   – Operationen auf Sprachen
   – Reguläre Sprachen

3. Spracherkennung

   – Entscheidbarkeit
   – Semientscheidbarkeit

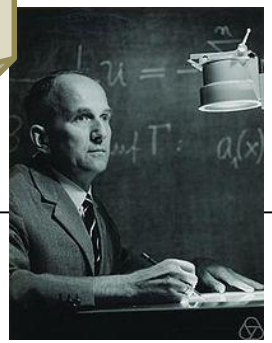
„Exotische“ Sprachen



- Sei π = 31415926…..
  - Alphabet $\Sigma := \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
  - $L_{pre\pi} := \{ w \in \Sigma^* | \, w \text{ ist Präfix von } \pi \}$
  - $L_{in\pi} := \{ w \in \Sigma^* | \, w \text{ ist Teilwort von } \pi \}$
- Fragen:
  - Gegeben ein $w \in \Sigma^*$
    - ist $w \in L_{pre\pi}$? — leicht zu beantworten
    - ist $w \notin L_{in\pi}$? — wie soll man das beantworten
      welche Möglichkeiten sind denkbar?

Collatz-Ulam-Sprache

Nur $k > 1$, sonst könnte die Funktion niemals terminieren.



- Alphabet
  $\Sigma := \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

- $L_{ulam} = \{n \in \Sigma^* - \{\varepsilon\} | \, ulam(n) = 1\}$
  - kann man feststellen, ob
    - $n \in L_{ulam}$?
    - $n \notin L_{ulam}$?
    - $L_{ulam} = \Sigma^* - \{\varepsilon\}$?

```
59    def ulam(n:BigInt){
60      var k=n;
61      while(k>1){
62        if(k%2==0) k=k/2
63        else k = k*3 + 1
64        println(k+" ")
65      }
66    }
67    def main(args: Array[String]): Unit = {
68      ulam(BigInt("12345667890087234578023089723457234705892375077302147492456
69      println
70    }
```

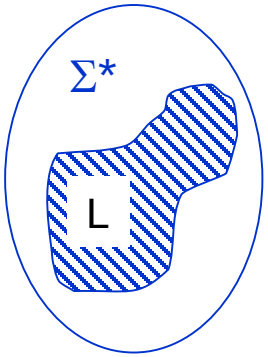
Tasks | Output - ScalaBeispiele (run) | Scala Console

```
617283394504361728901154486172861735294618753865107374622845678135371
185185018351308518670346345851858520588385626159532212386853703440611 4
925925091756542593351731729259292602941928130797661061934268517203057
277777527526962778005519518777787780882578439239298318580280555160917 2
138888763763481389002759759388893890441289219619649159290140277580458 6
694443818817406945013798796944469452206446098098245796450701387902293
208333145645222083504139639083340835661933829429473738935210416370688 0
104166572822611041752069819541670417830966914714736869467605208185344 0
520832864113055208760349097708352089154834573573684347338026040926720
260416432056527604380174548854176044577417286786842173669013020463360
130208216028263802190087274427088022288708643393421086834506510231680
651041080141319010950436372135440111443543216967105434172532551158 40
325520540070659505475218186067720055721771608483552717086266275579 20
162760270035329752737609093033860027860885804241776358543133137789 60
813801350176648763688045465169300139304429021208881792715665688948 0
406900675088324381844022732584650069652214510604440896357832844474 0
203450337544162190922011366292325034826107255302220448178916422237 0
101725168772081095461005683146162517413053627651110224089458211118 5
305175506316243286383017049438487552239160882953330672268374633355 6
152587753158121643191508524719243776119580441476665336134187316677 8
```

Entscheidbarkeit

Eine Sprache $L \subseteq \Sigma^*$ heißt entscheidbar, wenn es einen „Algorithmus“ E_L gibt, der zu jedem Input $w \in \Sigma^*$ entscheidet, ob $w \in L$ oder $w \in \Sigma^* - L$

- $w \in L \Leftrightarrow E_L(w) = \text{true}$
- $w \notin L \Leftrightarrow E_L(w) = \text{false}$



E_L berechnet i.W. die charakteristische Funktion von L:

$$\chi_L(w) = \begin{cases} 1, & \text{falls } w \in L \\ 0, & \text{falls } w \notin L \end{cases}$$

$L_{pre\pi}$ ist entscheidbar – wieso ?

Semi-Entscheidbarkeit

Eine Sprache $L \subseteq \Sigma^*$ heißt semi-entscheidbar, wenn es einen Algorithmus S_L gibt, der zu jedem Input $w \in \Sigma^*$ bestätigen kann, falls $w \in L$ ist.

$w \in L \Leftrightarrow S_L(w) = \text{true}$
$w \notin L \Leftrightarrow S_L(w) = \text{false}$ oder $S_L(w)$ terminiert nicht.

$L_{in\pi}$ ist semi-entscheidbar – wieso ?
L_{ulam} ist semi-entscheidbar – wie ?

Ist L_{ulam} entscheidbar ?
- Antwort bis heute unbekannt
- Vermutung: $L_{ulam} = \Sigma^* - \{\varepsilon\}$
  das würde bedeuten: ja

Entscheidbarkeit – Semi-Entscheidbarkeit

Satz: Eine Sprache $L \subseteq \Sigma^*$ ist entscheidbar
$\Leftrightarrow$ L und Σ^*-L sind semi-entscheidbar

- Ist $L \subseteq \Sigma^*$ entscheidbar, so ist L auch semi-entscheidbar
  - Klar: $S_L(w) = E_L(w)$
- Ist L entscheidbar, dann ist das Komplement Σ^*-L semi-entscheidbar
  - Klar: $S_{\Sigma^*-L}(w) := \text{not}(E_L(w))$
- Sind sowohl L als auch Σ^*-L semi-entscheidbar, dann ist L entscheidbar:
  - Lasse $S_L(w)$ und $S_{\Sigma^*-L}(w)$ parallel (oder quasiparallel unter Aufsicht eines schedulers) laufen.
  - Falls $w \in L$ hält $S_L(w)$ mit $S_L(w) = \text{true}$ // weil L semi-entscheidbar
  - Falls $w \notin L$ hält $S_{\Sigma^*-L}(w)$ mit $S_{\Sigma^*-L}(w) = \text{true}$ // weil Σ^*-L semi-entscheidbar

$$\mathbf{E_L}(w) = \begin{cases} \text{true}, & \text{falls } S_L(w) = \text{true} \\ \text{false}, & \text{falls } S_{\Sigma^*-L}(w) = \text{true} \end{cases}$$

Zusamenfassung

- Worte sind Listen von Zeichen
  - Induktiv definiert
  - Funktionen auf Worten rekursiv
- Sprachen sind Mengen von Worten
  - Kleene-Operationen $\circ, *, +$
  - Sonstige Operationen $\cap$, $-^R$, $-_e$
- Entscheidung, ob $w \in L$ kann schwer sein
  - Semi-entscheidbar:
    - falls $w \in L$ bestätigt der Algorithmus dies
  - Entscheidbar:
    - Algorithmus findet heraus ob $w \in L$ oder nicht
  - L entscheidbar $\Leftrightarrow$ L und $\sum^*-L$ semi-entscheidbar