Theoretische Informatik

Alphabete, Worte, Sprachen

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

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
  - Γ = {1} – *das unäre Alphabet*
  - Σ = { 0,1 } - *das Binäralphabet*
  - Σ = { - , . , □ } – das Morsealphabet (lang, kurz, Pause)
  - Γ = {0,1,...,9,A,B,C,D,E,F} – *das Hex-Alphabet*
  - Σ = {A,B,...,Z,a,...,z} – *das lateinische Alphabet*
  - Σ = {α, β, γ, ...} – das griechische Alphabet
  - Γ = { ๒, ๓, ๔, ๖, ๗, ๘, ๕, ๆ, ...} – Thai

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 2 Σ)
   - *v* das Restwort ist (v 2 Σ*)

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

- Das Zeichen ± lassen wir später meist weg.
  - In Java 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 $(a.\varepsilon) \pm u = a.u$
- Analog ua statt $u \pm a.\varepsilon$

Reverse und Palindrome

- Reverse
  - w^R ist das reverse Wort zu w
  - Formale Definition:
    1. $\varepsilon^R = \varepsilon$ // Fall: $w=\varepsilon$
    2. $(a.v)^R = v^R \pm (a.\ \varepsilon)$ // 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 $\Leftrightarrow$ $v = w \pm (a.\ \varepsilon)$ und w ist Palindrom

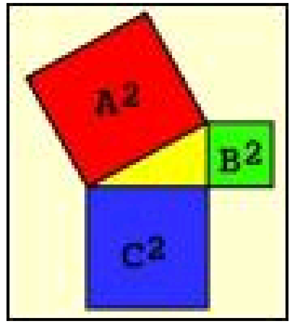
Teilworte, Präfixe, Suffixe

- u ist Präfix von w :⇔ ∃ v: u ± v = w.
  - ε präfix w ⇔ true
  - a.u präfix w ⇔ w=a.v ∧ u präfix v
- u ist Suffix von w :⇔ ∃ v: v ± u = w.
  - u suffix w ⇔ u^R präfix w^R
- t ist Teilwort von w :⇔ ∃ u,v: u ± t ± v = w.
  - u substring w ⇔ u präfix w ∨ w=a.v ∧ u substring 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. $8\ u,v,w\ 2\ \Sigma^*: (u \pm v) \pm w = u \pm (v \pm w)$ //assoziativ
  2. $8\ u,v\ 2\ \Sigma^*: |u \pm v| = |u| + |v|$
  3. $8\ w\ 2\ \Sigma^*: |w^R| = |w|$
  4. $8\ u,v\ 2\ \Sigma^*: (u \pm v)^R = v^R \pm u^R$
- Wie könnte man so etwas formal beweisen ?

Induktionsprinzip

- Aus der induktiven Definition von Σ^*:
  - ε 2 Σ^*
  - 8 a2Σ, 8 v2 Σ^*: a.v 2 Σ^*
- ergibt sich das Induktionsprinzip für Worte:
  - Sei P: Σ^* ! boolean eine „Wort-Eigenschaft“.

Aus

- P(ε)
- 8 a2 Σ: 8v2 Σ^*: P(v)) P(a¢v)

folgt

- 8 w2 Σ^*: P(w)

Induktionsanfang

Induktionsschritt

Induktionsschluss

Ein Induktionsbeweis

- Behauptung: 8 w2 Σ*: w ± ε = w
- Worteigenschaft : P(w) := w ± ε = w.
- Induktionsbehauptung : 8 w 2 Σ*: P(w), d.h. 8 w2 Σ*: w ± ε = w
  - Induktionsanfang:

    P(ε) , ε ± ε = ε , true // Fall 1 in Def von ±
  - Induktionsschritt:

    Wir müssen P(v)) P(a.v) beweisen

    Sei P(v), d.h. v ± ε = v // Induktionshypothese
    Es folgt (a.v) ± ε = a.(v ± ε) // Fall 2 in Def. von ±
    = a.v // Induktionshypothese
    Also ist P(a.v) bewiesen

    Also ist P(v)) P(a.v) bewiesen
- Induktionsschluss
  - 8 w2 Σ*: P(w).

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. 8 u,v2 Σ^*: |u ± v| = |u| + |v|
   Hinweis: Induktion über u, d.h.

   P(u) := 8 v2 Σ^*: |u ± v| = |u| + |v|

2. 8 u,v,w 2 Σ^*: (u ± v) ± w = u ± (v ± w)
   Hinweis: Induktion über u
   Verwendung des Lemmas: 8 u2 Σ^*: u ± ε = u

3. 8 w2 Σ^*: $|w^R|$ = |w|

4. 8 u,v2 Σ^*: $(u ± v)^R = v^R ± u^R$

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 µ Σ* ist eine (formale) Sprache
- Insbesondere sind auch Sprachen:
  - Σ* - die Menge aller Worte
  - ; = { } - die leere Menge
  - { ε } - die Menge, die das leere Wort enthält
  - { a } für a 2 Σ - ein-elementige Menge mit Wort der Länge 1

Beispiele formaler Sprachen

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

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

  wie kann man 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

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

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

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

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

Einige Gleichheiten

- L, R, S seien beliebige Sprachen. Es gelten z. B.:
  - $(L \cup R)\, S = LS \cup RS$
    - Beweis: $w \in (L \cup R)S \Rightarrow w = uv$ 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 : Alle Schritte sind umkehrbar
  - $S(RS)^* = (SR)^*S$
    - Beweis: $w \in S(RS)^* \Rightarrow \exists n \in N.\ \exists u_0,\ldots,u_n \in S\ \exists v_1,\ldots,v_n \in R.\ w = u_0(v_1u_1)\ldots(v_nu_n)$

    $\Rightarrow w = (u_0v_1)(u_1v_1)\ldots(u_{n-1} v_n)\, u_n$

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

Ausdrückbare Operationen

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

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

$$L^+ := L\,L^*$$

$$L? := L + \{\varepsilon\}$$

- Alle regulären Operatoren op sind monoton:
  - $L_1 \subseteq M_1$ und $L_2 \subseteq M_2 \quad \Rightarrow \quad L_1 \text{ op } L_2 \subseteq M_1 \text{ op } M_2$
  - $L \subseteq M \quad \Rightarrow \quad L^{op} \subseteq M^{op}$

Weitere Operationen

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

$L_1 \cap L_2$		Schnitt
$\Sigma^* - L$		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

Reguläre Sprachen

Sei Σ ein Alphabet. Reguläre Sprachen über Σ sind induktiv definiert

- Regulär sind:
  - ∅ // die leere Sprache
  - { ε } // enthält nur das leere Wort
  - { a } für jedes $a \in \Sigma$.
- Sind L und M reguläre Sprachen, dann auch
  - $L \cup M$
  - $L \pm M$
  - L^*

Reguläre Sprachen

Sei $\Sigma=\{a_1, a_2,\ldots, a_n\}$ ein Alphabet. Dann folgt, dass auch die folgenden Sprachen regulär sind:

- Σ - denn $\Sigma = \{a_1\} \cup \{a_2\} \cup \ldots \cup \{a_n\}$
- Σ^*
- $\{ w \}$ für jedes $w \in \Sigma^*$
  - denn für $w = w_1w_2\ldots w_n$ gilt $\{w\} = \{w_1\} \{w_2\}\ldots\{w_n\}$
- Jede endliche Teilmenge $L \subseteq \Sigma^*$

- Ist L regulär, dann auch
  - $L^n = LL \ldots L$ (n-mal) für jedes $n > 0$
  - $L^+ = LL^*$
  - $L? = L \cup \{\varepsilon\}$
  - $L^{\{m,n\}} = L^m (\{\varepsilon\} \cup L \cup L^2 \cup \ldots \cup L^{n-m})$
  - $L^{\{m,*\}} = L^m L^*$

„Exotische“ Sprachen

- Sei π = 31415927.....
  - Alphabet Σ := { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
  - $L_{pre\pi}$:= { w 2 Σ^* | w ist Präfix von π }
  - $L_{in\pi}$:= { w 2 Σ^* | w ist Teilwort von π }
- Fragen:
  - Gegeben ein w 2 Σ^*
    - ist w 2 $L_{pre\pi}$? — leicht zu beantworten
    - ist w 2 $L_{in\pi}$? — wie soll man das beantworten
welche Möglichkeiten sind denkbar ?

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

Ulam-Sprache

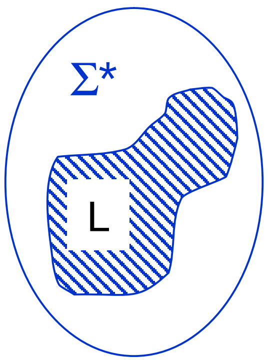
- Alphabet Σ := { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
  - $L_{ulam} = \{ n2\ \Sigma^*-\{\varepsilon\} \mid$ ulam(n) == true }
  - kann man feststellen, ob
    - $n \in L_{ulam}$?
    - $n \notin L_{ulam}$?
    - $L_{ulam} = \Sigma^*-\{\varepsilon\}$?

```
boolean ulam(int n){
      while(n>1){
        if(n%2==0) n=n/2;
        else     n=3*n+1;
      } return true;
}
```

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: $S_{\Sigma^*-L}(w) := not(E_L(w))$
- 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 $S_L(w) = true$ // weil L semi-entscheidbar
  - Falls $w \notin L$ hält $S_{\Sigma^*-L}(w)$ mit $S_{\Sigma^*-L}(w) = true$ // weil Σ^*-L semi-entscheidbar

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