Theoretische Informatik 2 bzw. Formale Sprachen und Berechenbarkeit

Sommersemester 2012

Prof. Dr. Nicole Schweikardt
AG Theorie komplexer Systeme

Goethe-Universität Frankfurt am Main

Herzlich willkommen!

Kapitel: Einführung

Worum geht's?

- Reguläre Sprachen:
  - Das stärkste "vollständig beherrschbare" Rechnermodell: Automatische Verifikation und Minimierung sind effizient möglich.
  - Verschiedene äquivalente Perspektiven:
    Reguläre Ausdrücke, reguläre Grammatiken,
    deterministische und nichtdeterministische endliche Automaten.
- Kontextfreie Sprachen:
  - Grundlage des Compilerbaus: effiziente Compiler
- Berechenbarkeit:
  - Was können Rechner heutiger und zukünftiger Technologien?
  - Welche algorithmischen Probleme sind (durch Rechner) lösbar, welche Probleme lassen sich nicht lösen?
- Die Chomsky-Hierarchie

Worauf wird aufgebaut?

- Diskrete Modellierung:
  - Beweismethoden
  - eine erste Behandlung von endlichen Automaten und kontextfreien Sprachen

- Datenstrukturen und GL-1 (Algorithmentheorie bzw. Theo. Inf. 1):
  - Laufzeitanalyse: O, o, Ω, ω and Rekursionsgleichungen
  - Traversierung von Graphen
  - Dynamische Programmierung
  - NP-Vollständigkeit

Literatur

- G. Schnitger, Skript zur Vorlesung „Formale Sprachen und Berechenbarkeit“ aus dem Sommersemester 2011, Goethe-Universität Frankfurt.
- J.E. Hopcroft, J.D. Ullman, Introduction to automata theory, languages and computation, Addison-Wesley, 1979.
- I. Wegener, Kompendium Theoretische Informatik – eine Ideensammlung, B.G. Teubner, 1996.
- I. Wegener, Theoretische Informatik: Eine algorithmenorientierte Einführung, B.G. Teubner 1999 (2. Auflage).
- U. Schöning, Theoretische Informatik - kurzgefasst, Springer 2001 (4. Auflage).
- M. Sipser, Introduction to the Theory of Computation, PWS Publishing, 1997.

Organisatorisches

- Die Webseite der Veranstaltung enthält alle wichtigen Informationen zur Veranstaltung wie Logbuch, Skript, Folien, Übungsblätter, Klausurtermine, usw.

  www.tks.informatik.uni-frankfurt.de/teaching/ss12/th-inf-2

- Übungsbetrieb: BITTE UNBEDINGT TEILNEHMEN!
  - Wöchentliche Ausgabe der Aufgabenblätter nach der Vorlesung
  - Abgabe der Lösungen: nach 1-wöchiger Bearbeitungszeit, vor Beginn der Vorlesung (frühere Abgabe auch im Büro von Joachim Bremer bzw. Frederik Harwath möglich)
  - Durch Bearbeiten der Übungsaufgaben können BONUSPUNKTE gesammelt werden.

Einige Grundbegriffe zum Thema Worte und Sprachen

Alphabete, Worte und Sprachen

1. $\mathbb{N} := \{0, 1, 2, 3 \ldots\}$ ist die Menge aller natürlichen Zahlen.
   $\mathbb{N}_{>0} := \{1, 2, 3, \ldots\}$ ist die Menge aller positiven natürlichen Zahlen.
2. Ein Alphabet Σ ist eine endliche, nicht-leere Menge von Buchstaben.
3. $\Sigma^n = \{a_1 \cdots a_n \mid a_1, \ldots, a_n \in \Sigma\}$ ist die Menge aller Worte der Länge n über Σ.
4. $\Sigma^0 = \{\varepsilon\}$ besteht nur aus dem leeren Wort ε.
5. $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$ ist die Menge aller Worte über dem Alphabet Σ.
6. $\Sigma^+ = \bigcup_{n \in \mathbb{N}_{>0}} \Sigma^n$ ist die Menge aller nicht-leeren Worte über Σ.
7. Für $w \in \Sigma^*$ ist $|w|$ die Länge von w, d.h. die Anzahl der Buchstaben in w.
   Für $a \in \Sigma$ ist $|w|_a$ die Anzahl der Vorkommen des Buchstabens a in w.

Eine Sprache L (über Σ) ist eine Teilmenge von Σ^*.

Beispiele für Sprachen

- Die Menge der im **Duden** aufgeführten Worte über dem Alphabet $\{a, A, \ldots, z, Z$,ä, Ä, . . . , ü,Ü, ß, $-\}$.
- **Deutsch** besteht aus allen syntaktisch korrekt aufgebauten und semantisch sinvollen Sätzen mit Worten aus dem Duden.
- **C++** ist die Menge aller syntaktisch richtig aufgebauten C++ Programme. Das Alphabet ist die Menge aller ASCII-Symbole.
- Die Sprache der **arithmetischen Ausdrücke** mit den Variablen x und y besteht aus allen arithmetischen Ausdrücken über dem Alphabet $\{x, y, +, *, -, /, (,)\}$.
- Weitere Beispielssprachen:
  - Die Menge aller **HTML-Dokumente**,
  - die Menge aller **XML-Dokumente**.

Operationen auf Sprachen

Sei Σ ein Alphabet, $u = u_1 \cdots u_n$ und $v = v_1 \cdots v_m$ seien Worte über Σ.

1. $uv = u_1 \cdots u_n \cdot v_1 \cdots v_m$ ist die Konkatenation von u und v.
2. Für Sprachen L_1, L_2 über Σ ist

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

   die Konkatenation von L_1 und L_2.
   Oft schreiben wir kurz $L_1 L_2$ oder $L_1 \cdot L_2$ statt $L_1 \circ L_2$.
3. Für eine Sprache L über Σ ist

$$\begin{aligned} L^n &= \{u_1 \cdots u_n \mid u_1, \ldots, u_n \in L\} \\ L^* &= \bigcup_{n \in \mathbb{N}} L^n \quad (\text{mit } L^0 := \{\epsilon\}) \end{aligned}$$

L^* ist die Kleene-Hülle (oder Kleene-Stern) von L.

Kompakte Beschreibung von Sprachen und Mengen

1. Die Menge aller Felder eines Schachbretts ist $\{A, B, C, D, E, F, G, H\} \circ \{1, 2, 3, 4, 5, 6, 7, 8\}$.
2. Die Menge aller Karten eines Skatblatts ist $\{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\} \circ \{7, 8, 9, 10, \text{Bube}, \text{Dame}, \text{König}, \text{Ass}\}$.
3. Die Menge der Binärdarstellungen der natürlichen Zahlen größer Null ist $\{1\} \circ \{0, 1\}^*$.
4. $\bigcup_{i=1}^{4} \{\cdot, -\}^i$ ist die Menge der Kodierungen von Buchstaben im Morsealphabet.
5. Die Menge der Uhrzeiten eines Tages ist $(\{2\} \circ \{0, 1, 2, 3\} \cup \{\epsilon, 1\} \circ \{0, 1, \ldots, 9\}) \circ \{:\} \circ \{0, 1, \ldots, 5\} \circ \{0, 1, \ldots, 9\}$.
6. Und die amerikanische Entsprechung ist $(\{1\} \circ \{0, 1, 2\} \cup \{1, \ldots, 9\}) \circ \{:\} \circ \{0, 1, \ldots, 5\} \circ \{0, 1, \ldots, 9\} \circ \{am, pm\}$.

Aus welchen Worten besteht die Sprache

$$K = (\{a\}^* \circ \{b\}^* \circ \{d\}^* \circ \{c\}^*)^*$$

- Behauptung: $K \subseteq \{a, b, c, d\}^*$
  - K besteht nur aus Worten über dem Alphabet $\{a, b, c, d\}$.
  - $\{a, b, c, d\}^*$ besteht aus allen Worten über $\{a, b, c, d\}$.
  - Behauptung stimmt.
- Behauptung: $\{a, b, c, d\}^* \subseteq K$
  - Der letzte $*$ in der Definition von K ist mächtig!
  - Sei $w = w_1 \cdots w_n$ ein beliebiges Wort in $\{a, b, c, d\}^*$:
    - Jeder Buchstabe gehört zu $(\{a\}^* \circ \{b\}^* \circ \{d\}^* \circ \{c\}^*)$.
    - Also ist $w = w_1 \cdots w_n \in (\{a\}^* \circ \{b\}^* \circ \{d\}^* \circ \{c\}^*)^n$
    - und deshalb ist $w \in (\{a\}^* \circ \{b\}^* \circ \{d\}^* \circ \{c\}^*)^* = K$.

Rekursive Definitionen von Sprachen

L sei die rekursiv wie folgt definierte Sprache über dem Alphabet $\Sigma = \{0, 1\}$:

Basisregel: $\epsilon \in L$

Rekursive Regel: wenn $u \in L$, dann $0u \in L$ und $u1 \in L$.

Wie sieht L aus?

(1) Behauptung: $\{0\}^* \{1\}^* \subseteq L$.

- Sei $w \in \{0\}^* \{1\}^*$, also $w = 0^n 1^m$ für $n, m \in \mathbb{N}$.
- Es ist $\epsilon \in L$ und deshalb $0\epsilon, 00\epsilon, \ldots, 0^n\epsilon = 0^n \in L$.
- Dann aber auch $0^n 1, 0^n 11, \ldots, 0^n 1^m \in L$
- und deshalb ist $w \in L$.

(2) Behauptung: $\{0\}^* \{1\}^* \subseteq L$.

- Dies sieht man leicht per Induktion nach dem Aufbau von L.

Somit ist $L = \{0\}^* \{1\}^*$.

Reguläre Ausdrücke

Die Menge der regulären Ausdrücke über einem endlichen Alphabet Σ wird rekursiv wie folgt definiert:

Basisregel: Die Ausdrücke $\emptyset$, ϵ und a für $a \in \Sigma$ sind regulär.

Die Ausdrücke stellen die leere Sprache ($L(\emptyset) = \emptyset$), die Sprache des leeren Wortes ($L(\varepsilon) = \{\varepsilon\}$) und die Sprache des einbuchstabigen Wortes a ($L(a) = \{a\}$) dar.

Rekursive Regeln: Sind R und S reguläre Ausdrücke, dann auch $(R \mid S)$, $(R \cdot S)$ und R^*.

$\mid$ ist die Vereinigung und wird manchmal auch mit $+$ bezeichnet ($L((R \mid S)) = L(R) \cup L(S)$),
$\cdot$ die Konkatenation und wird manchmal auch mit $\circ$ bezeichnet ($L((R \cdot S)) = L(R) \cdot L(S)$),
und $*$ ist die Kleene-Hülle ($L(R^*) = L(R)^*$).

Das Wortproblem

Das Wortproblem für eine Sprache L:

Für ein vorgegebenes Wort w, entscheide ob $w \in L$?

- Das Wortproblem für den Duden:
  Entscheide, ob ein vorgegebenes Wort im Duden ist.
  Einfach: Bitte nachschauen.
- Das Wortproblem für Deutsch:
  Entscheide, ob ein Satz syntaktisch richtig **und** sinnvoll ist.
  Sehr schwierig.
- Das Wortproblem für C++:
  Entscheide, ob ein C++ Programm syntaktisch korrekt ist.
  Compiler können das.

Wie schwer ist das Wortproblem für stets stets haltende C++ Programme?

Das Wortproblem für C++ Programme, die stets halten

```
main(int n)
int i;
{    i=n;
     while (NOT(prim(i) && prim(i+2)) )
        i++; }
```

- Die Funktion `prim(i)` gebe den Wert `wahr` genau dann zurück, wenn i eine Primzahl ist.
- Das Programm hält genau dann für eine Eingabe n, wenn es Primzahl-Zwillinge $i, i+2$ mit $i \geqslant n$ gibt.
- Das Programm hält genau dann immer, wenn es unendlich viele Primzahl-Zwillinge gibt. Leider ist die Frage, ob es unendlich viele Primzahl-Zwillinge gibt, bis heute offen!

Das Wortproblem für stets haltende C++ Programme ist extrem schwierig.

Das Wortproblem: Ein vorläufiges Fazit

Die Komplexität des Wortproblems variiert stark

- von trivialen Problemen wie dem Dudenproblem,
- zu ernst zunehmenden Problemen wie dem Compiler-Problem für C++ Programme,
- zu den sehr schwierigen NP-vollständigen Problemen,
- bis hin zu nicht lösbaren Problemen wie dem Wortproblem für stets haltende C++ Programme.

Wir betrachten später das Wortproblem für

- reguläre Sprachen (abhängig von der Repräsentation als determ. oder nichtdeterm. Automat oder als regulärer Ausdruck),
- kontextfreie Sprachen (wenn durch eine Grammatik repräsentiert)
- und auch für stets haltende C++ Programme.