Theoretische Informatik II

Einheit 6.4

Hierarchie von Komplexitätsklassen

1. Komplementäre Klassen
2. Polynomieller Platz
3. Logarithmischer Platz
4. Hierarchiesätze

Es gibt weitere wichtige Komplexitätsklassen

- ***co–$\mathcal{NP}$***
  - Probleme mit Komplement in $\mathcal{NP}$
  - Problem muß nicht notwendigerweise selbst in $\mathcal{NP}$ liegen
- $\boldsymbol{\Sigma_i^p}$ / $\boldsymbol{\Pi_i^p}$
  - Σ_2^p: Sprachen von OTMs, deren Orakel ein $\mathcal{NP}$-Problem entscheidet
  - Π_2^p: Sprachen von OTMs, deren Orakel ein *co*–$\mathcal{NP}$-Problem entscheidet
- ***PSPACE***
  - Platzverbrauch polynomiell relativ zur Größe der Eingabe
- ***LOGSPACE***
  - Platzverbrauch der Berechnung logarithmisch in Größe der Eingabe
- ***EXPTIME*** / ***EXPSPACE***
  - Rechenzeit und Platzverbrauch werden nicht mehr handhabbar

$co{-}\mathcal{NP}$: Probleme mit Komplement in $\mathcal{NP}$

$$co{-}\mathcal{C} := \{\, L \mid \overline{L} \in \mathcal{C} \,\}$$

- **Interessant für nichtdeterministische Klassen**
  - Für deterministische Komplexitätsklassen gilt $\mathcal{C} = co{-}\mathcal{C}$
    Akzeptieren/Verwerfen einer DTM ist vertauschbar
  - Nichtdeterministisches Akzeptieren ist komplizierter:
    OTM akzeptiert, wenn Orakel einen akzeptablen Lösungsvorschlag machen kann
- **Beispiele für Probleme in $co{-}\mathcal{NP}$:**
  - Menge der allgemeingültigen Formeln (Komplement von SAT)
  - Das Primzahlproblem (Komplement von Zusammengesetztheit)
- **Es gibt Probleme in $\mathcal{NP} \cap co{-}\mathcal{NP}$**
  - Das Primzahlproblem liegt auch in $\mathcal{NP}$ HMU, Satz 11.26
    Mittlerweile ist PRIMES $\in \mathcal{P}$ bekannt (Gilt $\mathcal{NP} \cap co{-}\mathcal{NP} = \mathcal{P}$?)
- **Sehr wahrscheinlich gilt $co{-}\mathcal{NP} \neq \mathcal{NP}$**
  - Wenn $\mathcal{P} = \mathcal{NP}$, dann $co{-}\mathcal{NP} = co{-}\mathcal{P} = \mathcal{P} = \mathcal{NP}$
  - $\mathcal{NP} = co{-}\mathcal{NP} \;\Leftrightarrow\; \mathcal{NPC} \cap co{-}\mathcal{NP} \neq \emptyset$
    $\Rightarrow$: offensichtlich, da $\mathcal{NPC} \neq \emptyset$
    $\Leftarrow$: Ist $L \in \mathcal{NPC} \cap co{-}\mathcal{NP}$ so gilt $\overline{L'} \leq_p L$ für jedes $L' \in co{-}\mathcal{NP}$ also $L' \leq_p \overline{L} \in \mathcal{NP}$

$PSPACE$: Polynomieller Platzverbrauch

- $\mathcal{NP} \subseteq PSPACE$
  – Eine Maschine kann in polynomieller Zeit nur polynomiell viele Speicherzellen verwenden
- $NSPACE(f(n)) \subseteq SPACE(f(n)^2)$ (Satz von Savitch)
  – Simulation mit Speicherung der Alternativen (§4.1) zu platzaufwendig
  – Teste $\kappa_\alpha \vdash^t \kappa_\omega$ (für maximales $t{=}2^{c*f(n)}$) durch "binäre Tiefensuche"
  – Platzverbrauch des Rekursionsstacks ist $\mathcal{O}(f(n)^2)$ (falls $f(n) \geq \log n$)
  – Zeitaufwand der Simulation ist exponentiell höher als bei der NTM
- $PSPACE = NPSPACE$ HMU §11.2.3
  – Folgt direkt aus dem Satz von Savitch
- $PSPACE \subseteq EXPTIME$
  – Eine terminierende Maschine, die maximal $f(n)$ Bandzellen aufsucht, terminiert nach maximal $c^{1+f(n)}$ Schritten
  – Für $|\Gamma| + |Q| = c$ sind maximal $c^{1+f(n)}$ Konfigurationen möglich

  $NPSPACE \subseteq NEXPTIME$ gilt aus dem gleichen Grund

$PSPACE$-Vollständigkeit

- **$\mathcal{C}$-Vollständigkeit allgemein**
  - L ist $\mathcal{C}$-vollständig, falls $L \in \mathcal{C}$ und $L' \leq_p L$ für alle $L' \in \mathcal{C}$
- **Wie zeigt man $PSPACE$-Vollständigkeit?**
  - Codiere Berechnungen von DTMs, die polynomiellen Platz brauchen
  - Sprache: komplexer als SAT, aber mit deterministischer Natur
  - Kandidat: Wahrheit geschlossener quantifizierter boolescher Formeln
- **Erweitere Aussagenlogik um boolesche Quantoren**
  - Ausagenlogische Variablen $P, Q, R, \ldots$, Konstante t und f
  - Formeln $\neg F$, $E \wedge F$, $E \vee F$, $E \Rightarrow F$, $(\forall P)F$, $(\exists P)F$
  - Wert von $(\forall P)F$ entspricht dem von $F[t/P] \wedge F[f/P]$
  - Wert von $(\exists P)F$ entspricht dem von $F[t/P] \vee F[f/P]$
  - Wert anderer Formeln wie in gewöhnlicher Aussagenlogik

    $(\forall P)(\exists Q)\,[(P \vee Q) \wedge (\neg P \vee \neg Q)]$ ist wahr (Wert 1)

    $(\exists Q)(\forall P)\,[(P \vee Q) \wedge (\neg P \vee \neg Q)]$ ist falsch (Wert 0)

  QBF ist die Menge der geschlossenen QB-Formeln mit Wert 1

QBF ist $PSPACE$-vollständig

- **$QBF \in PSPACE$**
  – Auswerten aussagenlogischer Formeln braucht linearen Platz
  – $(\forall P)F = F[t/P] \wedge F[f/P]$ und $(\exists P)F = F[t/P] \vee F[f/P]$ werden kaskadisch ausgewertet
  – Gesamtbedarf, einschließlich Zwischenspeicherung, ist quadratisch

- **Für alle $L \in PSPACE$ gilt $L \leq_p QBF$** HMU §11.3.4
  – Codiere Bandzellen und Konfigurationen wie im Satz von Cook
  – Beschreibe Formeln $F_{\kappa_1,\kappa_2,t}$ für die Aussage $\kappa_1 \vdash^t \kappa_2$
  – Zielformel ist $F_{\kappa_\alpha,\kappa_\omega,2^{c*p(n)}}$, wobei κ_α Anfangskonfiguration, κ_ω Endkonfiguration, $p(n)$ Platzverbrauch, c Alternativen pro Zelle
  – Setze $F_{\kappa_1,\kappa_1,0}$ und beschreibe $F_{\kappa_1,\kappa_2,1}$ passend zur Tabelle von δ
  – Beschreibe $F_{\kappa_1,\kappa_2,t}$ durch eine Darstellung für $(\exists\kappa)F_{\kappa_1,\kappa,t\div 2} \wedge F_{\kappa,\kappa_2,t\div 2}$, die das Entstehen exponentiell großer Formeln vermeidet
  $(\exists\kappa)(\forall\kappa_3)(\forall\kappa_4)[(\kappa_3 \Leftrightarrow \kappa_1 \wedge \kappa_4 \Leftrightarrow \kappa) \vee (\kappa_3 \Leftrightarrow \kappa \wedge \kappa_4 \Leftrightarrow \kappa_2)] \Rightarrow F_{\kappa_3,\kappa_4,t\div 2}$

Weitere *PSPACE*-vollständige Probleme

- **Strategische 2-Personen Spiele**
  - Viele konkrete Beispiele in Garey/Johnson Seite 254ff
  - Spielentscheidungen entsprechen alternierenden QB Formeln
    Spieler gewinnt, wenn für jeden Zug des Gegners, ein Zug existiert, so daß für jeden Folgezug des Gegners, ... das Resultat einen Sieg darstellt
  - QBF kann als strategisches Spiel beschrieben werden (und umgekehrt)
- **Sprache regulärer Ausdrücke**
  - Ist $L(E) = \Sigma^*$ für einen beliebigen regulären Ausdruck über Σ?
- **In-Place Acceptance** Asteroth/Baier §4.5
  - Kann eine gegebene DTM jedes Wort w ihrer Sprache mit Platzbedarf $|w|$ akzeptieren?

$\mathcal{LOGSPACE}$ Logarithmischer Platzverbrauch

- $\boldsymbol{LOGSPACE \subseteq \mathcal{P}}$
  – Gleiches Argument wie bei $PSPACE \subseteq EXPTIME$
- $\boldsymbol{LOGSPACE \stackrel{?}{=} NLOGSPACE}$ **ist ungeklärt**
  – Analyse benötigt Begriff der log-space Reduzierbarkeit
- **Es gibt $\boldsymbol{NLOGSPACE}$-vollständige Probleme**

  Sipser Satz 8.20

  – $\boldsymbol{PATH} = \{\, (G, v, v') \mid G{=}(V, E)$ gerichteter Graph $\wedge\ \exists\{v = v_1, .., v_n{=}v'\}{\subseteq}V.\ \ v_1..v_n$ Pfad in $G\,\}$
- $\boldsymbol{NLOGSPACE \subseteq \mathcal{P}}$

  Sipser Korollar 8.21

  – $PATH$ ist in polynomieller Zeit lösbar
- $\boldsymbol{NLOGSPACE = co{-}NLOGSPACE}$

  Sipser Satz 8.22

  – $\overline{PATH}$ liegt auch in $NLOGSPACE$

Wichtige Vertreter weiterer Komplexitätsklassen

- **Isomorphie ungerichteter Graphen** $\mathcal{NP}$, nicht vollständig
- **Zuverlässigkeit von Netzwerken** $\mathcal{NP}$-hart, vermutlich nicht in $\mathcal{NP}$
  – Wahrscheinlichkeit für fehlerfreie Verbindung zwischen zwei Knoten
- **Minimale äquivalente Schaltkreise** $\mathcal{NP}$-hart, nicht in $\mathcal{NP}$ (“Σ_2^p”)
  – Bestimme optimale Größe einer Schaltung
- **$n\times n$-Schach, Dame, Go**
  – Exponentiell viele Züge bis Spielende möglich $EXPTIME$(-vollständig)
- **TSP*: Bestimmung aller Rundreisen mit gegebenen Kosten**
  – Unrealistische Problemstellung: zu viele Lösungen $EXPSPACE$
- **Äquivalenz regulärer Ausdrücke mit Iteration**
  – Einfache Problemstellung mit sehr schwieriger Lösung
  – Ausdrücke dürfen $E^k = \underbrace{E\circ E...\circ E}_{k-mal}$ enthalten $EXPSPACE$-vollständig

Bedeutet mehr Zeit/Platz auch mehr lösbare Probleme?

- **Welche der folgenden Inklusionen sind echt?**

  $LOGSPACE \subseteq NLOGSPACE$
  $\subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq PSPACE = NPSPACE$
  $\subseteq EXPTIME \subseteq NEXPTIME \subseteq EXPSPACE \subseteq \ldots$

- **Wie beweist man Unlösbarkeit in Platz/Zeit f**
  - Diagonalisierung über alle Probleme, die in Komplexität f lösbar sind

- **Hilfsmittel: Konstruierbare Funktionen**
  - Funktionen, deren Komplexität durch ihren Wert beschränkt ist
  - $f : \mathbb{N} \to \mathbb{N}$ ist **platzkonstruierbar**, wenn $\hat{f}$ in Platz $\mathcal{O}(f)$ berechenbar
  - $f : \mathbb{N} \to \mathbb{N}$ ist **zeitkonstruierbar**, wenn $\hat{f}$ in Zeit $\mathcal{O}(f)$ berechenbar

    $\hat{f}$:$\{1\}^* \to \{0, 1, \}^*$ berechnet bei Eingabe 1^n die Binärdarstellung $r_b(f(n))$ von $f(n)$

    Rahmenbedingungen: $f(n) \geq \log n$ (Platz) bzw. $f(n) \geq n \log n$ (Zeit) für alle n
  - $\log n$, n^k, 2^n etc sind zeit- und platzkonstruierbar

- **Wichtiger formaler Begriff: Ordnung $o(f)$**
  - f als echte obere Schranke: $o(f) = \{g{:}\mathbb{N} \to \mathbb{R}^+ \mid \forall c>0.\ g <_a c * f\}$

Das Platzhierarchie-Theorem

Für jede platzkonstruierbare Funktion f gibt es ein $L \in SPACE(f)$ mit Platzkomplexität nicht in $o(f)$

- **Konstruiere L durch Diagonalisierung**
  - Definiere L durch seine charakteristische Funktion
  - Sei $h(n) = \begin{cases} 1 & \Phi_i(n) \leq 2^{f(n)} \wedge s_{M_i}(n) \leq^{**} f(n) \wedge \varphi_i(n) = 0 \\ 0 & \text{sonst} \end{cases}$ mit $i = \pi_1(n)$

    $s_{M_i}(n) \leq^{**} f(n)$ $\equiv$ h benutzt zur Simulation von $\varphi_i(n)$ maximal Platz $f(n)$.
    Benutzt die Simulation von $\varphi_i(n)$ Platz $d * s_{M_i}(n)$, so muß $s_{M_i}(n) \leq f(n)/d$ gelten
  - h ist eine Entscheidungsfunktion, die in Platz $\mathcal{O}(f)$ berechenbar ist
  - Definiere $L := h^{-1}(\{1\})$ (also $\chi_L = h$), also $L \in SPACE(f)$
- **Die Platzkomplexität von L ist nicht in $o(f)$**
  - Falls L durch ein Programm mit Komplexität $o(f)$ entschieden wird, so gilt $\chi_L = \varphi_k$ für ein k mit $s_{M_k}(n) < c * f(n)$ für alle $c > 0$, $n \geq n_0$
  - Wähle $n := \langle k, n_0 \rangle$ (also $k = \pi_1(n)$) für das zu $(1/d)$ passende n_0
    Dann gilt $n \geq n_0$, also $s_{M_k}(n) < (1/d) * f(n)$ bzw $s_{M_i}(n) \leq^{**} f(n)$
  - Es folgt $n \in L \Leftrightarrow h(n) = 1 \Leftrightarrow \varphi_k(n) = 0 \wedge s_{M_i}(n) \leq^{**} f(n) \Leftrightarrow n \notin L$

Konsequenzen des Hierarchietheorems

- **Platzkomplexität bildet eine echte Hierarchie**
  - $SPACE(f) \subset SPACE(g)$ falls g platzkonstruierbar und $f \in o(g)$
  - $SPACE(n^{\epsilon}) \subset SPACE(n^{\epsilon'})$ für alle $0 \leq \epsilon < \epsilon'$
  - $NLOGSPACE \subseteq SPACE(\log^2 n) \subset SPACE(n) \subset PSPACE$
  - $NPSPACE \subset EXPSPACE$

- **Zeitkomplexität bildet eine echte Hierarchie**
  - **Für jede zeitkonstruierbare Funktion f gibt es ein $L \in TIME(f)$ mit Zeitkomplexität nicht in $o(f/\log f)$**
    - Beweis analog zu Platzhierarchietheorem
  - $TIME(f) \subset TIME(g)$ falls g platzkonstruierbar und $f \in o(g/\log g)$
  - $TIME(n^{\epsilon}) \subset TIME(n^{\epsilon'})$ für alle $0 \leq \epsilon < \epsilon'$
  - $\mathcal{P} \subset EXPTIME$

Komplexitätsklassenhierarchie