Deskriptive Komplexitätstheorie

Roland Axelsson

17. Januar 2005

Inhaltsverzeichnis

1 Einleitung

Die klassische Komplexitätstheorie beschäftigt sich mit der Frage, wie viel Speicherplatz oder Berechnungsschritte ein als Algorithmus formuliertes Problem bis zu seiner Lösung (üblicherweise durch eine Turing-Maschine) in Anspruch nimmt. Die dadurch entstehenden Hierarchiestufen stellen ein Mass für die Komplexität von Problemen dar.
Die deskriptive Komplexitätstheorie versucht im Gegensatz dazu, das Problem selbst mit Hilfe einer geeigneten, möglichst schwachen Logik zu beschreiben. Da unterschiedlich ausdrucksstarke Logiken zur Beschreibung verschiedener Probleme herangezogen werden müssen, lässt sich auch durch diese Betrachtungsweise eine Hierarchie für die Schwierigkeit von Problemen entwerfen.
Es stellte sich schliesslich heraus, dass exakte Korrespondenzen zwischen Elementen beider Hierarchien existieren.[1] Ihre Untersuchung ist nicht nur von theoretischem Interesse, sondern weist auch unmittelbaren praktischen Nutzen auf: Will man beispielsweise eine Anfragesprache für relationale Datenbanken definieren, stellt sich sofort die Frage, wie aufwändig die Implementierungen für ein gegebenes Ausdrucksmittel sind. Konkreter:
Eine Datenbank enthalte Informationen darüber, welche Fluglinien zwischen welchen Städten Direktverbindungen unterhalten. Dabei soll jede Fluglinie als eigene binäre Relation über einem Universum von Städten organisiert sein. Denkbare Anfragen wären:

- Ist es möglich, mit Fluglinie E von Stadt a nach Stadt b ohne Zwischenstop zu fliegen?

- Ist es möglich, mit Fluglinie E von Stadt a nach Stadt b zu fliegen?

- Ist es möglich, von Stadt a nach Stadt b zu fliegen und dabei genau eine Fluglinie zu verwenden?

Die Formulierung der ersten Anfrage ist offenbar nicht sehr schwierig: Eab. Wir benötigen hierzu Prädikatenlogik erster Stufe (abgekürzt FO für *First Order*). Die Antwort ist direkt aus einer Tabelle oder Relation auszulesen und man sieht leicht, dass der dazugehörige Algorithmus von geringer Komplexität ist (Suche im Array). Aber schon für die zweite Frage ist FO nicht mehr ausreichend. Wir benötigen einen rekursiven Operator, der die transitive Hülle von E auszudrücken vermag.
Wie wir sehen werden, kann man einen solchen Operator definieren und er korrespondiert mit einer Klasse von Problemen, die in NLOGSPACE liegen.
Die dritte Frage schliesslich erinnert stark an die zweite, mit dem Unterschied, dass sämtliche Tabellen (d.h. Relationen) zu befragen sind. Solche Quantifizierung über Relationen ist nur in Prädikatenlogik zweiter Stufe (SO) möglich. Diese Logik ist jedoch bereits so komplex, dass man nur Fragmente davon (mit eingeschränkten syntaktischen Konstrukten) betrachtet. Einige davon werden wir in Kapitel 3 vorstellen, nachdem ein paar grundsätzliche Begriffe definiert

[1] erstmals 1974 entdeckt von Fagin durch das Ergebnis NPTIME $\equiv \Sigma_1^1$ [2]

wurden.
Im Anschluss daran werden Grundkenntnisse aus der Komplexitätstheorie aufgefrischt (Kapitel 4) bevor wir in Kapitel 5 das grundsätzliche Verfahren und ein paar mehr oder weniger technische Details der deskriptiven Komplexitätstheorie angeben. Schliesslich werden zwei Ergebnisse ausführlich untersucht, nämlich dass FO(DTC) $\equiv$ LOGSPACE und dass FO(TC) $\equiv$ NLOGSPACE. Andere interessante Ergebnisse werden nur kurz im letzten Kapitel erwähnt.

2 Begriffe

2.1 Notation

Zu logischen Formeln φ werden oft ihre (freien sowie gebundenen) Variablen in der Form $\varphi(x_1, ..., x_k)$ mit angegeben. Kleinbuchstaben bezeichnen hierbei Individuenvariablen, Grossbuchstaben Relationsvariablen. Variablentupel werden durch $\bar{x}$ abgekürzt, wenn die Länge keine Rolle spielt.
Sei $\varphi(a_1, ..., a_k)$ eine Formel und $t_1, ..., t_n$ Terme. $\varphi\frac{t_1,...,t_n}{a_1,...,a_n}$ oder kürzer $\varphi(t_1, ..., t_n)$ bedeutet, dass in φ alle freien Variablenvorkommen von $a_1, ..., a_n$ durch $t_1, ..., t_n$ ersetzt werden.
Belegungen einer Formel φ werden in eckigen Klammern angegeben: $\varphi[\alpha]$.

2.2 Vokabular, Struktur

Definition 2.1 Ein *Vokabular* ist eine endliche, nichtleere Menge τ von Relationen- und Konstantensymbolen. Jedem Relationssymbol ist eine natürliche Zahl n zugeordnet, seine sogenannte Stelligkeit. τ heisst *relational*, falls es keine Konstantensymbole enthält.

Definition 2.2 Eine *Struktur* $\mathcal{A}$ eines Vokabulars τ besteht aus einer nichtleeren Menge A (*Universum, Wertebereich*), einer n-stelligen Relation R^A über A für jedes n-stellige Relationssymbol R aus τ und aus einem Element c^A aus A für jedes Konstantensymbol c aus τ.

2.3 Ordnung

Definition 2.3 Sei $\tau = \{<\}$ ein Vokabular und $\mathcal{A}$ eine τ-Struktur $\mathcal{A} = (A, <^A)$. $\mathcal{A}$ heisst *Ordnung*, falls für alle $a, b, c \in A$:

(1) nicht $a <^A a$
(2) $a <^A b$ oder $b <^A a$ oder $a = b$
(3) wenn $a <^A b$ und $b <^A c$ dann $a <^A c$

Manchmal betrachten wir endliche Ordnungen $\mathcal{A}$ als $\{<, S, min, max\}$-Strukturen, wobei S die binäre Nachfolgerrelation ist und min, max das erste bzw. letzte Element der Ordnung ist. $\mathcal{A}$ ist eine Ordnung, falls zusätzlich zu (1),(2),(3) folgende Bedingungen für alle $a, b \in A$ gelten:

(4) $S^A ab$ gdw ($a <^A b$ und für alle c, wenn $a <^A c$ dann $b <^A c$ oder $b = c$)
(5) $min^A <^A a$ oder $min^A = a$
(6) $a <^A max^A$ oder $max^A = a$

2.4 Disjunkte Vereinigung

Definition 2.4 Seien $\mathcal{A}, \mathcal{B}$ τ-Strukturen mit $\mathcal{A} \cap \mathcal{B} = \emptyset$. $\mathcal{A} \dot\cup \mathcal{B}$ ist die τ-Struktur mit Universum $A \cup B$ und $R^{\mathcal{A} \dot\cup \mathcal{B}} := R^A \cup R^B$ für jedes R in τ.
Falls $A \cap B \neq \emptyset$ nehme man isomorphe Kopien $\mathcal{A}'$ und $\mathcal{B}'$ von $\mathcal{A}$ bzw. $\mathcal{B}$ mit disjunkten Universen.
NB: Die Vereinigung von geordneten Strukturen ist keine geordnete Struktur.

3 Fixpunktlogiken

Rekursivität ist in FO nicht mehr ausdrückbar. Man betrachte dazu beispielsweise gerichtete Graphen $\mathcal{G} = (G, E^G)$. Das Erreichbarkeitsproblem, also ob Knoten y von Knoten x via E^G zu erreichen ist, kann in FO nicht formuliert werden.[2] Zwar kann zu jedem festen Graphen eine Formel angegeben werden, deren Erfüllungsmenge genau das gewünschte leistet, aber das allgemeine Problem für beliebige Graphen ist nicht mehr definierbar.

$$
\begin{aligned}
&\varphi_1 := Exy \\
&\varphi_2 := \varphi_1 \vee \exists z(Exz \wedge Ezy) \\
&\varphi_3 := \varphi_2 \vee \exists u \exists v(Exu \wedge Euv \wedge Evy) \\
&\qquad\qquad\qquad \vdots
\end{aligned}
$$

Für jedes $i \in \mathbb{N}$ lässt sich auf diese Weise eine Formel konstruieren, die das Erreichbarkeitsproblem in i Schritten beschreibt. Falls die Anzahl der Knoten i nicht übersteigt, kann man davon ausgehen, dass die Formel für ein beliebiges Knotenpaar x, y das Problem über die Mitgliedschaft in der Erfüllungsmenge entscheidet.
Wollte man eine allgemeine Formel für beliebige Graphen in FO angeben, müsste sie unendliche Länge und unendlich viele gebundene Variablen im Fokus der Existenzquantoren haben, die beliebig grosse Knotenmengen „verkraften“ könnten, was nicht zulässig ist.[3]
Stattdessen kann man das Erreichbarkeitsproblem in SO formulieren:

$$Xxy \vee \exists z(Xxz \wedge Xzy)$$

[2] Dieses Problem entspricht der zweiten Datenbankanfrage im einleitenden Kapitel

[3] Intuition: Jede endliche Formel ψ besitzt höchstens n Variablen. Wähle Graphen mit Knotenanzahl $n+1$ und maximalem endlichem Pfad $a_0 a_1 ... a_n a_{n+1}$ (paarweise verschieden). Offensichtlich ist das Erreichbarkeitsproblem für a_0 und a_{n+1} nicht durch ψ entscheidbar. Also muss ψ unendlich sein.

Da zweitstufige Logik jedoch ein viel zu starkes Ausdrucksmittel ist, behilft man sich mit Fragmenten davon, die nun vorgestellt werden sollen.

3.1 Fixpunkte

Sei M eine endliche nichtleere Menge. Eine Funktion $F : \mathcal{P}(M) \to \mathcal{P}(M)$ induziert eine Folge $\emptyset, F(\emptyset), F(F(\emptyset)), ...$ von Teilmengen von M.
Ihre Elemente werden mit $F_0, F_1, ...$ bezeichnet. Also ist $F_0 = \emptyset$ und $F_{n+1} = F(F_n)$. Man überlegt sich leicht, dass falls es ein $n_0 \geq 0$ gibt, so dass $F_{n_0+1} = F_{n_0}$, dann gilt $F_m = F_{n_0}$ für alle $m \geq n_0$.
Wir bezeichnen F_{n_0} als *Fixpunkt* von F und schreiben dafür F_∞. Falls F_∞ nicht existiert, setzen wir $F_\infty := \emptyset$.
F heisst *inflationär*, falls für alle $X \subseteq M : X \subseteq F(X)$.
Zu beliebigem F kann eine inflationäre Funktion $F' : \mathcal{P}(M) \to \mathcal{P}(M)$ mit $F(X) := X \cup F(X)$ definiert werden.

Wir definieren nun eine Funktion $F^\varphi : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$, welche die Erfüllbarkeitsmenge von φ bezüglich einer Relation R angibt. Sei $\varphi(x_1, ..., x_k, \bar{u}, X, \bar{Y})$ eine Formel[4] aus dem Vokabular τ, wobei die Relationsvariable X Stelligkeit k habe. Sei $\mathcal{A}$ eine τ-Struktur, $\bar{b}$ eine Interpretation von $\bar{u}$ in A und $\bar{S}$ eine Interpretation von $\bar{Y}$ über A.

$$F^\varphi(R) := \{(a_1, ..., a_k) | \mathcal{A} \models \varphi[a_1, ..., a_k, \bar{b}, R, \bar{S}]\}$$

Ein Beispiel zur Illustration: Sei $\mathcal{G} = (G, E^G)$ ein Graph und

$$\varphi(x, y, X) := (Exy \vee \exists z(Xxz \wedge Ezy))$$

Dann ist für $n \geq 1$

$$F_n^\varphi = \{(a, b) | \text{es gibt einen Pfad der Länge} \leq n \text{ von } a \text{ nach } b\}^5$$

und mithin

$$F_\infty^\varphi = \{(a, b) | \text{es gibt einen Pfad von } a \text{ nach } b\}$$

Wir werden F^φ im Anschluss benötigen, um eine Semantik für FO(IFP), FO(LFP) bzw. FO(PFP) geben zu können. Aber zunächst sei noch die Syntax kurz vorgestellt.

3.2 FO(IFP), FO(LFP) und FO(PFP)

Wir erweitern nun FO um Operatoren IFP, LFP bzw. PFP, mit deren Hilfe sich Rekursivität ausdrücken lässt. IFP steht für inflationärer Fixpunkt, LFP für kleinster Fixpunkt (engl. *least*) und PFP schliesslich für partieller Fixpunkt. Die Syntax von FO(IFP) sei wie folgt induktiv gegeben:

---

[4] Zur besseren Lesbarkeit im Folgenden: $\bar{u}$ bzw. $\bar{Y}$ bezeichnen hier Vektoren von erst- bzw. zweitstufigen Konstantensymbolen

[5] das n im Subskript bedeutet n-fache Anwendung von F auf sich selbst

$$\varphi := \ p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x\varphi \mid [\mathrm{IFP}_{\bar{x},X}\varphi]\bar{t}$$

wobei p eine atomare SO-Formel über einem Vokabular τ ist und $\bar{x}$ und $\bar{t}$ dieselbe Länge besitzen. Man erhält die Syntax von FO(PFP), indem man das I aus IFP in obigem Ausdruck durch ein P ersetzt und die Syntax von FO(LFP), indem man stattdessen ein L verwendet und ausserdem verlangt, dass X in φ nur positiv, d.h. nicht negiert vorkommt.[6]

Die Semantik der FO-Ausdrücke wird als bekannt vorrausgesetzt. Die Bedeutung von $[\mathrm{IFP}_{\bar{x},X}\varphi]\bar{t}$ ist $\bar{t} \in F_\infty^{(X\bar{x}\vee\varphi)}$ und diejenige von $[\mathrm{PFP}_{\bar{x},X}\varphi]\bar{t}$ ist $\bar{t} \in F_\infty^\varphi$ (Wir erinnern uns aus Kapitel 3.1, dass $F^{(X\bar{x}\vee\varphi)} = (F^\varphi)'$ und deshalb auch tatsächlich inflationär ist). $[\mathrm{LFP}_{\bar{x},X}\varphi]\bar{t}$ ist wahr, wenn $\bar{t} \in F_{n_0}^\varphi = F_{n_0+1}^\varphi$ und n_0 die kleinste Zahl ist, für die diese Bedingung gilt. Präziser: Falls X k-stellig ist und die freien Variablen aus $[\mathrm{IFP}_{\bar{x},X}\varphi]\bar{t}$ (bzw. [LFP...], [PFP...])[7] aus $\bar{u}$ und $\bar{Y}$ sind, und $\bar{b}$ und $\bar{S}$ Interpretationen von $\bar{u}$ und $\bar{Y}$ in $\mathcal{A}$ sind, dann

$$\mathcal{A} \models [\mathrm{IFP}_{\bar{x},X}\varphi]\bar{t}[\bar{b},\bar{S}] \text{ gdw } (t_1[\bar{b}],...,t_k[\bar{b}]) \in F_\infty^{(X\bar{x}\vee\varphi)}$$

$$\mathcal{A} \models [\mathrm{LFP}_{\bar{x},X}\varphi]\bar{t}[\bar{b},\bar{S}] \text{ gdw } (t_1[\bar{b}],...,t_k[\bar{b}]) \in F_{n_0}^\varphi = F_{n_0+1}^\varphi \text{ und für alle } m < n_0, \\ F_m^\varphi \neq F_{m+1}^\varphi$$

$$\mathcal{A} \models [\mathrm{PFP}_{\bar{x},X}\varphi]\bar{t}[\bar{b},\bar{S}] \text{ gdw } (t_1[\bar{b}],...,t_k[\bar{b}]) \in F_\infty^\varphi$$

Betrachten wir erneut das Beispiel der Graphen:

$$\varphi(x,y) := [\mathrm{IFP}_{x,y,X} Exy \vee \exists z(Xxz \wedge Ezy)]xy$$

drückt aus, dass x und y durch einen Pfad verbunden sind. Also wäre die Klasse der verbundenen Graphen in FO(IFP) axiomatisierbar durch $\forall x \forall y(\neg(x=y) \rightarrow \varphi(x,y))$ und ein paar Graphenaxiomen aus FO.

3.3 FO(DTC) und FO(TC)

Die beiden Logiken, die nun vorgestellt werden sollen, sind ebenfalls Erweiterungen von FO, nämlich um Operatoren DTC bzw. TC von englisch *deterministic transitive closure* bzw. *transitive closure* was übersetzt *deterministische transitive Hülle*, bzw. nur *transitive Hülle* heisst. Die Syntax von FO(DTC) ist gegeben durch:

$$\varphi := \ p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x\varphi \mid [\mathrm{DTC}_{\bar{x},\bar{y}}\varphi]\bar{s}\bar{t}$$

wobei p eine atomare FO-Formel über einem Vokabular τ ist und alle Variablen in $\bar{x}$ und $\bar{y}$ paarweise verschieden sind und dieselbe Länge wie $\bar{s}$ und $\bar{t}$ besitzen. Ersetze DTC durch TC und erhalte die Syntax von FO(TC).

Die Semantik der Logiken ist mit Hilfe folgender Operationen definiert:
Sei R eine binäre Relation auf einer Menge M, $R \subseteq M \times M$.

[6] dadurch wird erreicht, dass F^φ monoton ist, weshalb der kleinste Fixpunkt existiert

[7] bestehend aus den freien Variablen aus $\bar{t}$ vereinigt mit den freien Variablen aus φ ohne $\bar{x}$ und X

$$\mathrm{DTC}(R) := \{(a,b) \in M \times M |\text{ es gibt } n > 0 \text{ und } e_0, ..., e_n \in M \text{ so dass } a = e_0, b = e_n, \text{ und für alle } i < n, \text{ ist } e_{i+1} \text{ das eindeutige } e, \text{ für welches } (e_i, e) \in R\}$$

$$\mathrm{TC}(R) := \{(a,b) \in M \times M |\text{ es gibt } n > 0 \text{ und } e_0, ..., e_n \in M \text{ so dass } a = e_0, b = e_n, \text{ und für alle } i < n, (e_i, e_{i+1}) \in R\}$$

Die Bedeutung von $[\mathrm{DTC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y},\bar{u})]\bar{s}\bar{t}$ ist also

$$(\bar{s},\bar{t}) \in \mathrm{DTC}(\{(\bar{x},\bar{y})|\varphi(\bar{x},\bar{y},\bar{u})\}),$$

und die Bedeutung von $[\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y},\bar{u})]\bar{s}\bar{t}$ ist

$$(\bar{s},\bar{t}) \in \mathrm{TC}(\{(\bar{x},\bar{y})|\varphi(\bar{x},\bar{y},\bar{u})\}),$$

wobei $\{(\bar{x},\bar{y})|\varphi(\bar{x},\bar{y},\bar{u})\}$ als binäre Relation auf der Menge der Tupel der Länge $\bar{x}$ des Universums betrachtet wird.

Das Problem der verbundenen Graphen lässt sich auch in FO(TC) ausdrücken: $\forall x \forall y(\neg(x = y) \rightarrow [TC_{x,y}Exy]xy)$.

3.4 Ausdrückbarkeit

Seien $\mathcal{L}_1$ und $\mathcal{L}_2$ Logiken. $\mathcal{L}_1 \leq \mathcal{L}_2$ (lies: $\mathcal{L}_1$ ist höchstens so ausdrucksstark oder mächtig wie $\mathcal{L}_2$) falls es für jedes Vokabular τ und für jede Formel $\varphi \in \mathcal{L}_1[\tau]$ eine Formel $\psi \in \mathcal{L}_2[\tau]$ gibt, so dass $\mathrm{Mod}(\varphi) = \mathrm{Mod}\ (\psi)$.

Wir zeigen nun eine Aussage über die Mächtigkeitshierarchie der Logiken aus dem vorangegangenen Kapitel:[8]

$$\mathrm{FO(DTC)} \leq_{(1)} \mathrm{FO(TC)} \leq_{(2)} \mathrm{FO(IFP)} =_{(3)} \mathrm{FO(LFP)} \leq_{(4)} \mathrm{FO(PFP)}$$

(1) Wir müssen offenbar eine Einschränkung auf lineare Strukturen vornehmen, um den Determinismus auch für den TC-Operator zu garantieren:
$\models [\mathrm{DTC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y},\bar{u})]\bar{s}\bar{t} \leftrightarrow [\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y},\bar{u}) \wedge \forall \bar{z}(\varphi(\bar{x},\bar{z},\bar{u}) \rightarrow \bar{z} = \bar{y})]\bar{s}\bar{t}$

(2) Der TC-Operator ist nur für binäre Relationen definiert, weshalb dem IFP-Operator eine binäre Relationsvariable zugeführt werden muss:
$\models [\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y},\bar{u})]\bar{s}\bar{t} \leftrightarrow [\mathrm{IFP}_{\bar{x}\bar{y},X}\varphi(\bar{x},\bar{y},\bar{u}) \vee \exists \bar{v}(X\bar{x}\bar{v} \wedge \varphi(\bar{v},\bar{y},\bar{u}))]\bar{s}\bar{t}$

(3) Der Beweis für dieses Ergebnis ist länglich. Siehe [3] für Details.

(4) Wir erinnern uns an den Kniff aus der Semantikdefinition des IFP-Operators, welcher einer Formel durch Anhängen einer Disjunktion der Relationsvariable X garantiert, dass der Fixpunkt inflationär ist (also $X \subseteq F(X)$). Dies können wir natürlich auch explizit tun:
$\models [\mathrm{IFP}_{\bar{x},X}\varphi]\bar{t} \leftrightarrow [\mathrm{PFP}_{\bar{x},X}(X\bar{x} \vee \varphi)]\bar{t}$

[8] diese gilt wohlgemerkt nur auf endlichen Strukturen

4 Komplexitätstheorie

Das wichtigste in der Komplexitätstheorie verwendete Berechenbarkeitsmodell ist die Turing-Maschine. Ihr grundsätzliches Konzept wird als bekannt vorausgesetzt, wir beschreiben lediglich die hier verwendete Variante.

4.1 Maschinenmodell

Wir betrachten mehrbändige Turing-Maschinen mit einseitig unendlichen Bändern. Auf jedem Band befindet sich ganz links am begrenzten Ende des Bands eine Zelle, die mit einem (nicht zum Alphabet gehörigen) Zeichen α beschriftet ist, welches nicht überschrieben werden kann. Die Zellen auf den Bändern sind durchnummeriert, beginnend mit -1 bei der α-Zelle. Bestimmte Bänder sind als Eingabebänder reserviert und nicht beschreibbar, sondern nur lesbar. Diese Bänder haben am rechten Ende ihres Eingabestrings das (nicht zum Alphabet gehörige) Zeichen ω stehen.[9] Die übrigen Bänder heissen Arbeitsbänder.
Die Maschine besitzt einen Startzustand s_0, und ausgezeichnete Zustände s_+ (akzeptierender Endzustand) und s_- (verwerfender Endzustand), in denen die Berechnung anhält.
Eine Turing-Maschine heisst *deterministisch*, falls es für jeden Zustand und für jedes Alphabetszeichen höchstens eine Anweisung gibt. Ein Eingabewort u wird von einer Turing-Maschine M akzeptiert, falls es einen Durchlauf gibt, in dem M im Zustand s_+ anhält. M verwirft u, falls alle Durchläufe in s_- halten.

4.2 Zeit- und Platzkomplexität

Sei $\mathbb{A}$ ein Alphabet. Ist $f : \mathbb{N} \to \mathbb{N}$ eine Funktion, so heisst eine Turing-Maschine M *f zeitbeschränkt*, falls es für jedes Eingabewort $w \in \mathbb{A}$ einen endlichen Durchlauf von M gibt, der höchstens $f(|w|)$ Schritte benötigt.
M heisst *f platzbeschränkt*, falls es für alle Eingabeworte $w \in \mathbb{A}$ einen endlichen Durchlauf gibt, der höchstens $f(|w|)$ Zellen auf jedem Arbeitsband benötigt.
Sei $\mathbb{N}[x]$ die Menge der Polynome mit Koeffizienten aus $\mathbb{N}$. Eine Sprache $L \subseteq \mathbb{A}^+$ ist in PTIME („Menge der in polynomieller Zeit erkennbaren Sprachen") bzw. in PSPACE („Menge der mit polynomiellem Platzbedarf erkennbaren Sprachen"), wenn sie von einer deterministischen Turing-Maschine akzeptiert wird und p zeitbeschränkt, bzw. p platzbeschränkt ist, wobei p ein Polynom aus $\mathbb{N}[x]$ ist.
NPTIME und NPSPACE sind analog dazu definiert, sie erlauben allerdings zusätzlich nichtdeterministische Turing-Maschinen.
Weitere Komplexitätsklassen, die uns hier beschäftigen werden, sind LOGSPACE und NLOGSPACE, die mit logarithmischer Platzschranke offenbar unterhalb des Platzbedarfs des Eingabewortes liegen. Um die sublinearen Komplexitätsklassen überhaupt einführen zu können, zählt man deshalb den Platzbedarf der Eingabebänder nicht mit hinzu.
Eine Sprache L ist in NLOGSPACE, falls es eine nichtdeterministische Turing-Maschine M und ein $d \geq 0$ gibt, so dass L von M akzeptiert wird und $d{*}\log$

[9] zur Vereinfachung der Instruktionen

platzbeschränkt ist. Ist M deterministisch, so ist $L \in$ LOGSPACE.

Es gilt offensichtlich:

$$\text{PTIME} \subseteq \text{NPTIME} \quad \text{und}$$

$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{PSPACE} \subseteq \text{NPSPACE}$$

und man kann zeigen, dass

$$\text{NPTIME} \subseteq \text{PSPACE} \quad \text{und} \quad \text{PSPACE} = \text{NPSPACE}$$[10]

Insgesamt gilt also

$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE}$$

$$= \text{NPSPACE}$$

5 Komplexitätsklassen und logische Axiomatisierbarkeit

Die deskriptive Komplexitätstheorie untersucht den Zusammenhang zwischen algorithmischer Komplexität und logischer Axiomatisierbarkeit. Es stellt sich heraus, dass in Bezug auf bestimmte Klassen von Problemen die zu ihrer Formulierung benötigte Ausdrucksstärke einer Logik mit der Berechnungsschranke einer für diese Problemklasse konstruierten Turing-Maschine korrespondiert. Das Bindeglied zwischen beiden Gebieten stellen die Strukturen dar, welche einerseits eine bestimmte Formel einer bestimmten Logik erfüllen und andererseits (geeignet kodiert) genau die innerhalb fester Zeit- oder Platzschranken akzeptierten Eingabeworte einer bestimmten Turing-Maschine sind.
Auf diese Weise können Ergebnisse aus dem Bereich der Komplexitätstheorie auf die Logik übertragen werden und umgekehrt. Beispielsweise könnte man das PTIME=NPTIME Problem lösen, indem man nachweist, dass FO(IFP) und Σ_1^1 dieselbe Mächtigkeit besitzen, weil man weiss, dass $\Sigma_1^1 \equiv$ NPTIME und dass FO(IFP) $\equiv$ PTIME (eine Übersicht und Angaben darüber, wo sich die Resultate nachschlagen lassen, findet sich in Kapitel 6.2).[11]
Sei K eine Klasse geordneter τ-Strukturen und $\mathcal{L}$ eine Logik. Wir interessieren uns für die folgende gegenseitige Charakterisierung von $\mathcal{L}$ und $\mathcal{C}$:

Es gibt TM M, die K akzeptiert und Komplexität $\mathcal{C}$ benötigt	$\Leftrightarrow$	K in $\mathcal{L}$ axiomatisierbar, d.h. es gibt Formel $\varphi \in \mathcal{L}$, so dass $K = \text{Mod}(\varphi)$[12]

[10] siehe [5]

[11] Σ_1^1 ist ein Fragment von SO, das aus Formeln der Form $\exists X_1 ... \exists X_m \psi$ besteht, wobei ψ aus FO ist

Wir erläutern zunächst ganz allgemein das angewandte Verfahren, wie man zu diesen Charakterisierungen kommen kann und geben im Anschluss einige der Ergebnisse an, wobei das Beispiel NLOGSPACE $\equiv$ FO(TC) ausführlich behandelt wird. Kapitel 5.1 beschäftigt sich mit der „$\Rightarrow$"-Richtung, Kapitel 5.2 mit der Richtung „$\Leftarrow$".

5.1 Kodierung von Strukturen

Wir werden im folgenden nur Turing-Maschinen betrachten, deren Eingabewörter endliche Strukturen sind und uns dafür interessieren, welchen Komplexitätsklassen sie angehören. Da Turing-Maschinen auf Strings arbeiten, werden Stringrepräsentationen von Strukturen benötigt. Strukturen sind jedoch abstrakte Objekte (ebenso wie ihre Elemente) und deshalb gibt es für sie keine kanonische Repräsentation. Natürlich soll eine Maschine aber für jede Repräsentation dieselben Resultate liefern. Dieses Problem kann hier umschifft werden, da es für unsere Überlegungen ausreichend ist, sich im Bereich der geordneten Strukturen zu bewegen, für die man eine (isomorphe) kanonische Repräsentation angeben kann.
Sei $\mathcal{A}$ eine geordnete Struktur mit $|A| = n$. Wir bilden eine isomorphe Kopie von $\mathcal{A}$, indem wir als Universum $A = \{0, ..., n-1\}$ und dazu die natürliche Ordnung $<^A$ annehmen.
Sei $\mathcal{A}$ eine $\{<, S, min, max\}$-Struktur[13] und $\mathcal{B}$ eine Struktur auf dem Vokabular $\tau = \{R_1, ..., R_k, c_1, ..., c_l\}$, $\mathcal{C} = \mathcal{A}\dot{\cup}\mathcal{B}$. Bei der Kodierung von $\mathcal{C}$ wird die Ordnungsrelation nicht direkt auf die Eingabebänder geschrieben, sondern implizit verwendet.
Wir benötigen $1 + k + l$ Eingabebänder und m Arbeitsbänder für ein $m \geq 1$. Das verwendete Alphabet besteht nur aus dem Zeichen „1". „0" steht anstelle des üblichen „blank" auf dem Band, d.h. fast alle Zellen eines Bandes enthalten „0".
Auf dem nullten Band befindet sich eine Folge von „1", der Anzahl der Elemente des Universums entsprechend.
Die nächsten k-Bänder enthalten Informationen über die Relationen R_i, wobei $1 \leq i \leq k$. Sei R_i r-stellig und $|C| = n$. Wir betrachten zu jeder Relation eine $(n^r \times r)$-Matrix. Jede Zeile der Matrix entspricht einer möglichen Kombination von A^r.[14] Die Zeilen seien lexikographisch geordnet. Auf dem i-ten Band wird nun für jedes $q \in R_i \subseteq A^r$ eine „1" auf die j-te Zelle geschrieben, falls q in der j-ten Zeile der Matrix kodiert ist.
Ein Beispiel: Sei $A = \{0, 1\}$ Universum einer Struktur $\mathcal{A}$ und enthalte $\mathcal{A}$ die Relation $R_i = \{(0,0,1), (0,1,0), (1,0,0)\}$. Die lexikographisch geordnete Matrix sähe so aus:

[13] siehe Kapitel 2.3
[14] natürlich ergibt sich die Zeilenanzahl aus $|A|^r = n^r$

$$\begin{array}{cccc}
 & 0 & 0 & 0 \\
\rightarrow & 0 & 0 & 1 \\
\rightarrow & 0 & 1 & 0 \\
 & 0 & 1 & 1 \\
\rightarrow & 1 & 0 & 0 \\
 & 1 & 0 & 1 \\
 & 1 & 1 & 0 \\
 & 1 & 1 & 1
\end{array}$$

R_i ist in den Zeilen 1,2 und 4 kodiert. Also enthält das i-te Band der Turing-Maschine die folgenden Einträge:

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\
\hline
\alpha & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & \omega \\
\hline
\end{array}$$

Für $1 \leq i \leq l$ wird das $k+i$-te Eingabeband mit der binären Repräsentation von c_i^A beschrieben.

5.2 Axiomatisierbarkeit von Strukturen

Sei K eine Klasse von τ-Strukturen und $\mathcal{L}$ eine Logik. K ist *axiomatisierbar* in $\mathcal{L}$, falls es eine Formel φ aus $\mathcal{L}$ mit Vokabular τ gibt, so dass $K = \mathrm{Mod}(\varphi)$, d.h. K enthält genau die Strukturen, die φ erfüllen.

Ebenso, wie Strukturen kodiert werden müssen, damit sie in kanonischer Weise von Turing-Maschinen verarbeitet werden können, stellt sich in der anderen Beweisrichtung das Problem, dass Konfigurationen von Turing-Maschinen als Relation $CONF$ angegeben werden müssen, damit sie sich axiomatisieren lassen. Diese (allgemeingültige) Kodierung ist recht aufwändig, aber ihre Details sind hier nicht relevant, weil sich für die Beweise der folgenden Kapitel eine vergleichsweise einfache Kodierung findet. Dies ist darauf zurückzuführen, dass sich Konfigurationen, die sich auf Bänder mit sublinearem Platz beschränken, unabhängig von der Länge des Eingabewortes als n-Tupel angeben lassen. Es sei an dieser Stelle nur erwähnt, dass sich die Kodierung für höhere Komplexitätsklassen schwieriger erweist. Für näheres siehe [1].

6 Ergebnisse der Deskriptiven Komplexitätstheorie

In diesem Kapitel sollen einige der Ergebnisse der deskriptiven Komplexitätstheorie vorgestellt werden. Ein Grossteil wird dabei nur kurz erwähnt, lediglich die Beweise für LOGSPACE/NLOGSPACE und FO(DTC)/FO(TC) werden ausführlich dargestellt.

6.1 NLOGSPACE $\equiv$ FO(TC) und LOGSPACE $\equiv$ FO(DTC)

6.1.1 „$\Rightarrow$“

Grundüberlegung:
Sei M eine Turing-Maschine, die eine Klasse K von Strukturen in NLOGSPACE akzeptiert. Aufgabe ist es, eine Formel φ aus FO(TC) zu konstruieren, so dass $K = \mathrm{Mod}(\varphi)$.
Wir wissen, dass M die Struktur $\mathcal{A}$ akzeptiert, genau dann wenn es eine Folge von ($d{*}\log(|A|)$-platzbeschränkten) Konfigurationen $\bar{a}_0, ..., \bar{a}_k$ gibt, so dass $\bar{a}_0$ die Startkonfiguration, $\bar{a}_{i+1}$ die Nachfolgekonfiguration von $\bar{a}_i$ und $\bar{a}_k$ eine akzeptierende Konfiguration ist. Das heisst, hätten wir eine Formel $\chi_{succ}(\bar{x}, \bar{x}')$, welche ausdrückt, dass $\bar{x}'$ eine ($d{*}\log(|A|)$-platzbeschränkte) Nachfolgekonfiguration von $\bar{x}$ ist, könnten wir mit Hilfe des TC-Operators „berechnen“, ob es einen Pfad von der Startkonfiguration zu einer akzeptierenden Konfiguration gibt. Genauer:

$$M \text{ akzeptiert } \mathcal{A}$$
$$\Leftrightarrow$$
$$\mathcal{A} \models \exists \bar{x}(\chi_{start}(\bar{x}) \wedge \exists \bar{x}'([\mathrm{TC}_{\bar{x},\bar{x}'}\chi_{succ}(\bar{x}, \bar{x}')]\bar{x}, \bar{x}' \wedge x'_1 = s_+))$$

wobei $\chi_{start}(\bar{x})$ eine Formel aus FO sein soll, die ausschliesslich bei Belegung mit der Startkonfiguration erfüllt ist.
Bevor wir die Formeln $\chi_{start}(\bar{x})$ und $\chi_{succ}(\bar{x}, \bar{x}')$ angeben können, müssen wir noch die Kodierung der Konfigurationen betrachten:
Wie es die Schreibweise einer Konfiguration als Tupel $\bar{x}$ in obigen Formeln bereits andeutet, lassen sich $d{*}\log(n)$-platzbeschränkte Konfigurationen tatsächlich ohne Verrenkungen als Tupel über dem Universum auffassen. Um die Sache nicht unnötig zu verkomplizieren, nehmen wir an, dass die auf M kodierte Struktur $\mathcal{A}$ lediglich aus einer Relation R bestehe, ausserdem Kardinalität $n > d{*}\log(n)$ habe und n grösser als die Anzahl der Zustände von M sei. M habe nur ein Arbeitsband.
Das folgende Tupel von Elementen aus $A = \{0, ..., n-1\}$ leistet das gewünschte:

$$(z, u_\alpha, u_\omega, u, v_\alpha, v_\omega, v_0, v_1, w_\alpha, w, y_0, ..., y_d)$$

wobei

- z der Zustand ist (eine Zahl $< n$)
- u_α, u_ω, u die Position des Lesekopfes auf dem 0-ten Band (das „Universumsband“) kodieren (hier reicht eine Variable aus A nicht, denn wir haben Länge $n + 2$):

$$u_\alpha := \begin{cases} 0 & \text{falls der Kopf nicht } \alpha \text{ liest} \\ n-1 & \text{falls der Kopf } \alpha \text{ liest} \end{cases}$$

$$u_\omega := \begin{cases} 0 & \text{falls der Kopf nicht } \omega \text{ liest} \\ n-1 & \text{falls der Kopf } \omega \text{ liest} \end{cases}$$

u ist die Position der Zelle, die vom Kopf gescannt wird, falls es eine innere ist, ansonsten ist $u = 0$.

- $v_\alpha, v_\omega, v_0, v_1$ die Kopfposition auf dem Band für die binäre Relation R ganz ähnlich kodieren; $v_0 * n + v_1$ ist dabei die Zellennummer

- w_α, w die Position des Kopfes auf dem Arbeitsband darstellen. Da M in NLOGSPACE ist, wird jedes Wort auf dem Arbeitsband maximale Länge $< \log(n)$ haben. Deshalb kann die Position des Schreib-/Lesekopfs auf dem Arbeitsband als Variable $< n$ angegeben werden.

- $y_0 ... y_d$ die Konkatenation der Inschrift auf dem Arbeitsband ist (für die Inschrift selbst reicht natürlich nicht unbedingt ein Element aus A, denn jede der d geschriebenen Variablen ist aus A).

Wir schreiben ab jetzt für das Konfigurationstupel nur noch $\bar{x}$. Nun können die Teilformeln $\chi_{start}(\bar{x})$ und $\chi_{succ}(\bar{x}, \bar{x}')$ wie folgt angegeben werden:

$$\begin{aligned}&\chi_{start}(\bar{x}) := \bar{x} = (0, ..., 0)\\&\chi_{succ}(\bar{x}, \bar{x}') := \chi_{acc}(\bar{x}, \bar{x}') \vee \bigvee \chi_{instr}(\bar{x}, \bar{x}')\end{aligned}$$

wobei

$$\chi_{acc}(\bar{x}, \bar{x}') := (x_1 = s_+ \wedge \bar{x} = \bar{x}')$$

und wo für jeden Maschinenbefehl

$$\text{instr} = sb_0b_1c_1 \to s'c_1'h_0h_1h_2$$

$\chi_{instr}(\bar{x}, \bar{x}')$ eine Formel ist, die falls $\bar{x}$ eine $d*\log$ platzbeschränkte Konfiguration ist, ausdrückt, dass $\bar{x}$ die Basis $sb_0b_1c_1$ hat, die Nachfolgekonfiguration von $\bar{x}$ bezüglich instr $d*\log$ platzbeschränkt und $\bar{x}'$ ist.
Beispiel: Sei instr $= s1\alpha 1 \to s'0(-1)11$,[15] dann ist $\chi_{instr}(\bar{x}, \bar{x}')$ die Konjunktion aus

$z = s$
"s ist der aktuelle Zustand"
$u_\alpha = 0 \wedge u_\omega = 0$
"der Kopf des 0-ten Bandes befindet sich in einer inneren Zelle"
$v_\alpha = n - 1 \wedge v_\omega = 0 \wedge v_0 = 0 \wedge v_1 = 0$
"der Kopf des ersten Bandes scannt α"
$w_\alpha = 0 \wedge \text{One}_d\ y_0, ..., y_d w$
"der Kopf des Arbeitsbands scannt eine 1"
$z' = s'$
"s' ist der neue Zustand"
$u'_\omega = 0 \wedge ((u > 0 \wedge Su'u \wedge u'_\alpha = 0) \vee (u = 0 \wedge u' = 0 \wedge u'_\alpha = n - 1))$[16]
"neue Kopfposition des 0-ten Bandes"

[15] -1, 0, 1 soll für die Kopfbewegungen links, garnicht, rechts bedeuten
[16] S bezeichnet hier die Nachfolgerrelation auf A

$v'_\alpha = 0 \wedge v'_0 = 0 \wedge v'_1 = 0 \wedge v_\omega = 0$
"die Kopfposition auf dem ersten Band ist Zelle 0"
$w'_\alpha = 0 \wedge Sww' \wedge w' < d * log(n)$
"die neue Position des Arbeitsbands ist innerhalb der Schranke"
$\text{Zero}_d\ y'_0 ... y'_d w$
"die neue Inschrift der Zelle auf dem Arbeitsband ist gescannt"
$\text{Equal}_d\ y_0 ... y_d w y'_0 ... y'_d$
"die Inschrift bei nicht gescannten Zellen unverändert".

Fehlen noch die Definitionen von Zero_d, One_d und Equal_d. Sei $l = \log n - 1$.

$\text{Zero}_d\ \bar{u}k$	$\Leftrightarrow$	$\bar{u} < 2^l, k < (d+1) * l$, und die k-te Ziffer der Konkatenation $u_0 ... u_d$ ist 0
$\text{One}_d\ \bar{u}k$	$\Leftrightarrow$	$\bar{u} < 2^l, k < (d+1) * l$, und die k-te Ziffer der Konkatenation $u_0 ... u_d$ ist 1
$\text{Equal}_d\ \bar{u}k\bar{u}$	$\Leftrightarrow$	$\bar{u}, \bar{u}' < 2^l, k < (d+1) * l$, und die Worte $u_0, ..., u_d$ und $u'_0, ..., u'_d$ unterscheiden sich höchstens an der k-ten Stelle

6.1.2 „$\Leftarrow$"

Sei $K \in$ FO(TC), d.h. es gibt eine Formel φ aus FO(TC), so dass für alle Strukturen $\mathcal{A}$, $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{A} \in K$. Wir zeigen per Induktion über den Aufbau von φ, dass es eine Turing-Maschine M gibt, die *stark bezeugt*, dass $K \in$ NLOGSPACE, d.h.,
- M akzeptiert K
- für jede geordnete Struktur $\mathcal{A}$ hält jeder Durchlauf von M, der mit $\mathcal{A}$ gestartet wurde in s_+ oder s_-
- jeder Durchlauf von M erfüllt die Bedingung der logarithmischen Platzschranke

Sei φ eine atomare Formel, der Einfachheit halber sogar $\varphi = Rxy$. Es soll gezeigt werden, dass

$$\{(\mathcal{A}, i, j) | R^A ij\} \in \text{(N)LOGSPACE}$$

Wenn $\mathcal{A}$ gemäss Kapitel 5.1 kodiert wurde, dann befindet sich die Information, ob $R^A ij$ gilt, in der $(i * n + j)$-ten Zelle des Eingabebandes, das R entspricht. Die Maschine führt diese Berechnung durch (ohne (N)LOGSPACE zu verlassen) und überprüft dort, ob φ erfüllt ist.

$\varphi = \neg\psi$: An dieser Stelle tritt das Problem auf, dass ψ einen TC-Operator enthalten könnte. Im Vergleich zu dem Fall, dass der TC-Operator nicht im Scope eines Negationszeichens liegt und M in jedem Durchlauf nur eine transitive Hülle berechnen muss, um die Erfüllbarkeit von φ zu bezeugen, scheint dieser Fall recht ungünstig: Wird der TC-Operator negiert, so müssen sämtliche transitiven Hüllen berechnet werden, damit der bezeugende Fall ausgeschlossen werden kann. Tatsächlich kann man jedoch zeigen, dass es zu jeder Formel aus FO(TC) eine äquivalente Formel aus FO(posTC) gibt. In FO(posTC) darf der TC-Operator nur im Bereich einer geraden Anzahl Negationszeichen vorkom-

men.[17]
Somit bereitet uns die Konstruktion von M keine Schwierigkeiten mehr: Die Induktionsvoraussetzung erlaubt es uns, von einer Maschine M' auszugehen, die in NLOGSPACE ist und die geordneten Modelle von ψ stark bezeugt. Vertausche die Rollen von s_+ und s_-.

$\varphi(x_1, ..., x_k) = \psi_1 \vee \psi_2$: Wir schalten die Maschinen M_{ψ_1} und M_{ψ_2} für die Teilformeln ψ_1 und ψ_2 (die es wegen Induktionsvoraussetzung gibt) hintereinander und akzeptieren, falls eine der beiden Maschinen akzeptiert. Die Arbeitsbänder werden nach der Berechnung von ψ_1 wieder gelöscht (falls die Berechnung von ψ_2 noch nötig sein sollte).

$\varphi(x_1, ..., x_k) = \exists x\psi$: Die Maschine M für φ nimmt die Maschine M_0 für $\psi(x_1, ..., x_{k-1}, x)$ (gegeben durch Induktionsvoraussetzung) und testet alle Elemente des Universums als Belegungen für x. Dazu muss jedes Element des Universums in binärer Repräsentation sukzessive auf das Arbeitsband geschrieben werden. Falls der Test mindestens einmal gelingt, hält M in s_+, sonst in s_-.

$\varphi = [\mathrm{TC}_{\bar{x},\bar{y}}\psi]\bar{s}\bar{t}$: Nehmen wir der Einfachheit halber an, dass $\bar{x} = x, \bar{y} = y, \bar{s} = s$ und $\bar{t} = t$. Per Induktionsvoraussetzung haben wir eine Maschine M_0, die stark bezeugt, dass die geordneten Modelle von ψ in NLOGSPACE sind. Falls es einen „Pfad“ von s nach t gibt (in einer gegebenen Struktur $\mathcal{A}$), so hat er maximale Länge $n := |A|$. Die Maschine M für φ benötigt einen Zähler, der eine Subroutine höchstens n Mal aufruft (d.h. M verwirft andernfalls). $i := s$ wird auf ein Arbeitsband geschrieben. Die Subroutine rät nichtdeterministisch ein Element j des Universums und testet unter Zuhilfenahme von M_0, ob $\psi[i, j]$ gilt. Falls nicht, dann hält M in s_-. Andernfalls muss überprüft werden, ob $j = t$, wobei im positiven Fall M akzeptiert und im negativen $i := j$ gesetzt wird.
Hier der Algorithmus in Pseudocode:

```
i:=s;
FOR z:=0..n-1{
  j:=CHOOSE(0..n-1);
  IF(NOT ψ[i,j]) THEN REJECT;
  ELSE IF(j = t) THEN ACCEPT;
  i:=j;
}
REJECT;
```

Wir benötigen offenbar 3*log(n) zusätzliche Speicherzellen für die Werte von i, j und z, weshalb wir NLOGSPACE nicht verlassen.
Zusammengenommen haben wir gezeigt, dass $K \in \mathrm{FO(TC)} \Rightarrow K \in$ NLOGSPACE.
Der Beweis für $K \in \mathrm{FO(DTC)} \Rightarrow K \in$ LOGSPACE ist bis auf den letzten Fall identisch mit obigem. Der Algorithmus hierfür sei kurz in Pseudocode gegeben:

[17] Man hat auch gezeigt, dass NLOGSPACE=coNLOGSPACE [4]

```
i:=s;
FOR z:=0..n-1{
  IF(ψ[i,j] for exactly one j < n) THEN {
    IF(j = t) THEN ACCEPT;
    ELSE i:=j;
  }
  ELSE REJECT;
}
REJECT;
```

Es ist klar, dass auch mit diesem Algorithmus die Bedingungen von LOGSPACE nicht verletzt werden.

6.2 Übersicht über die anderen Ergebnisse

Für geordnete Strukturen gilt:

FO(DTC)	$\equiv$	LOGSPACE	
FO(TC)	$\equiv$	NLOGSPACE	
FO(IFP)	$\equiv$	PTIME	siehe [1]
Σ^1_1	$\equiv$	NPTIME	siehe [2]
FO(PFP)	$\equiv$	PSPACE	siehe [1]

Literatur

[1] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Perspectives in Math. Logic. Springer, Berlin, 1995.

[2] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity and Computation*, 7:43–73, 1974.

[3] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[4] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

[5] W. J. Savitch. Deterministic simulation of nondeterministic Turing Machines. In *Symp. on Theory of Computing, STOC'69*, pages 247–248, New York, May 1969. ACM.