# 14 Die polynomielle Hierarchie von Larry Stockmeyer

## 14.1 Einleitende Betrachtungen

Wir gehen der Frage nach, ob die Klasse der Sprachen, welche (in einem zu präzisierenden Sinne) reduzierbar auf eine Sprache aus *NP* sind, mit *NP* zusammenfällt oder *NP* echt enthält. Im zweiten Fall würde die Reduktionstechnik gewissermaßen für eine „neue Qualität“ an Rechenkraft sorgen. Es wird sich zeigen, dass die Antwort auf diese Frage vom gewählten Reduktionsbegriff abhängt. Im Falle von Cook-Reduktionen entsteht eine Komplexitätsklasse die (voraussichtlich) *NP* echt enthält, wohingegen polynomielle Reduktionen (= Karp-Reduktionen) nicht aus *NP* hinausführen.

### 14.1.1 Polynomielle Reduktionen auf Sprachen aus NP

Falls $L \leq_{pol} L'$ und $L' \in NP$, dann folgt bekanntlich $L \in NP$. Wir erhalten nämlich eine polynomielle NTM $M$ für $L$ mit Hilfe der polynomiellen NTM $M'$ für $L'$ und der polynomiellen Reduktion $f : \Sigma^* \to \Sigma^*$ wie folgt:

1. Transformiere die Eingabe $x$ zum Mitgliedschaftsproblem für die Sprache $L$ in die korrespondierende Eingabe $x' = f(x)$ zum Mitgliedschaftsproblem für die Sprache $L'$.
2. Sezte $M'$ auf $x'$ an und akzeptiere genau dann, wenn $M'$ akzeptiert.

Die Klasse der auf Sprachen aus *NP* polynomiell reduzierbaren Sprachen führt also nicht aus *NP* heraus.

### 14.1.2 Cook-Reduktionen auf Sprachen aus NP

Bei Cook-Reduktionen ergibt sich ein völlig anderes Bild. Wir wollen dies durch ein paar Beobachtungen und Beispiele illustrieren.

Eine erste Beobachtung wäre, dass jede Sprache auf ihr Komplement Cook-reduzierbar ist:

**Beobachtung 14.1** $\forall L \subseteq \Sigma^* : L \leq_T \bar{L}$.

**Beweis** Zu Eingabe $x$ können wir das $\bar{L}$-Orakel befragen und seine Antwort umkehren. **qed.**

Aus Beobachtung 14.1 ergibt sich unmittelbar, dass jede Sprache aus $NP \cup \text{co-}NP$ auf das Erfüllbarkeitsproblem Cook-reduzierbar ist:

**Beobachtung 14.2** $\forall L \in NP \cup co\text{-}NP : L \leq_T \text{SAT}$.

**Beweis** Für $L \in NP$ ergibt sich $L \leq_{pol} \text{SAT}$ und somit auch $L \leq_T \text{SAT}$ aus der *NP*-Vollständigkeit von SAT. Für $L \in \text{co-}NP$ folgt zunächst $\bar{L} \in NP$. Zusammen mit Beobachtung 14.1 erhalten wir die Reduktionskette $L \leq_T \bar{L} \leq_T \text{SAT}$. Nun folgt $L \leq_T \text{SAT}$ aus der Transitivität von Cook-Reduktionen. **qed.**

Die Klasse der auf SAT Cook-reduzierbaren Sprachen führt also (unter der $NP \neq$ co-$NP$-Voraussetzung) aus der Klasse $NP$ heraus. Eine DOTM mit „Rechenkraft“ $P$ und ein Orakel der „Rechenkraft“ $NP$ können demzufolge in Kooperation eine „Rechenkraft“ oberhalb von $NP$ erlangen.

Ein Orakel für eine $NP$-vollständige Sprache nennen wir im folgenden einfach $NP$-Orakel. Wir wollen illustrieren, wie mächtig eine polynomielle NOTM wird, wenn sie mit einem $NP$-Orakel ausgestattet ist. Dabei spielen die in den folgenden Beispielen beschriebenen Sprachen eine wichtige Rolle:

**Beispiel 14.3** *MINIMUM EQUIVALENT CIRCUIT (kurz: MEC) sei die Sprache aller Paare (genauer: Kodierungen von Paaren) der Form $(S, k)$, wobei gilt:*

- *$S$ ist ein Boolescher Schaltkreis.*
- *$k$ ist eine nicht-negative ganze Zahl.*
- *Es gibt einen Booleschen Schaltkreis $S'$ mit höchstens $k$ Bausteinen (aus der Basis AND, OR, NOT), der die gleiche Boolesche Funktion wie $S$ berechnet.*

*Das zu MEC korrespondierende Minimierungsproblem gehört in den Bereich der Hardwareminimierung: suche die billigste Realisierung eines gegebenen Schaltkreises.*

**Beispiel 14.4** *UNSAT (ausgeschrieben: UNSATISFIABILITY) sei die Sprache der nicht erfüllbaren Booleschen CNF-Formeln $C$. Da UNSAT im Wesentlichen das Komplement von SAT ist, ist UNSAT co-NP-vollständig.*

**Beispiel 14.5** SAT$^*$ *sei die Sprache aller Schaltkreise (genauer: Kodierungen von Schaltkreisen), die nicht die konstante Nullfunktion berechnen. Da SAT offensichtlich ein Teilproblem von* SAT$^*$ *ist und andererseits* SAT$^*$ *zu NP gehört, ist* SAT$^*$ *NP-vollständig.*

Wir werden im Folgenden nachweisen, dass MEC ein sehr hartes Problem ist: es ist co-$NP$-hart unter polynomiellen Reduktionen und $NP$-hart unter Cook-Reduktionen. Vermutlich gehört es nicht zu $NP \cup$co-$NP$, aber diese Frage ist offen. Unter der $NP \neq$ co-$NP$-Voraussetzung kann man zumindest ausschließen, dass MEC zu $NP$ gehört. Der Härte von MEC zum Trotz werden wir eine polynomielle NOTM angeben, die mit Hilfe eines $NP$-Orakels als Akzeptor von MEC auftritt. Die kombinierte Rechenkraft zweier Agenten (NOTM und Orakel) mit „Rechenkraft“ $NP$ reicht also aus, um die „harte Nuss“ namens MEC zu knacken.

Wir starten unsere Analyse von MEC mit der folgenden

**Beobachtung 14.6** *$UNSAT \leq_{pol} MEC$.*

**Beweis** Wir verwenden im Prinzip die Technik der Spezialisierung. Sei $C$ eine Boolesche CNF-Formel, $f_C$ die von $C$ berechnete Boolesche Funktion und $S_C$ der zweistufige Schaltkreis, der $C$ auf die offensichtliche Weise berechnet (Berechnung der Klauseln mit OR-Gattern auf der ersten Stufe und Berechnung der Konjunktion aller Klauseln mit einem

AND-Gatter auf der zweiten Stufe). OBdA enthalte $C$ mindestens eine nicht-tautologische Klausel.[1] Dann gilt erst recht $f_C \not\equiv 1$ und die folgenden Äquivalenzen sind offensichtlich:

$$\begin{aligned} C \in \text{UNSAT} &\Leftrightarrow f_C \equiv 0 \\ &\Leftrightarrow f_C \text{ ist eine konstante Funktion} \\ &\Leftrightarrow f_C \text{ ist von einem Schaltkreis mit 0 Bausteinen berechenbar} \\ &\Leftrightarrow (S_C, 0) \in \text{MCE} \end{aligned}$$

Da $(S_C, 0)$ leicht aus $C$ berechenbar ist, erhalten wir eine polynomielle Reduktion von UNSAT auf MCE. **qed.**

**Folgerung 14.7** *1.* SAT $\leq_T$ *MEC.*

*2. MEC ist co-NP-hart unter polynomiellen oder Cook-Reduktionen.*

*3. MEC ist NP-hart unter Cook-Reduktionen.*

*4. MEC* $\notin$ *NP unter der NP* $\neq$ *co-NP-Voraussetzung.*

**Beweis**

1. Aus UNSAT $\leq_{pol}$ MEC (gemäß Beobachtung 14.6) folgt UNSAT $\leq_T$ MEC. Wegen SAT $\leq_T$ UNSAT (gemäß Beobachtung 14.1) ergibt sich dann SAT $\leq_T$ MEC aus der Transitivität der Cook-Reduktionen.

2. Die co-*NP*-Härte von UNSAT vererbt sich wegen UNSAT $\leq_{pol}$ MEC auf MEC.

3. Die *NP*-Härte von SAT (unter polynomiellen Reduktionen) impliziert die *NP*-Härte von SAT unter Cook-Reduktionen. Diese vererbt sich wegen SAT $\leq_T$ MEC auf MEC.

4. Würde MEC als ein (unter polynomiellen Reduktionen) co-*NP*-vollständiges Problem zu *NP* gehören, dann würde jede Sprache $L \in$ co-*NP* wegen $L \leq_{pol}$ MEC und MEC $\in$ *NP* ebenfalls zu *NP* gehören. (Vgl. Abschnitt 14.1.1.) Dies hätte co-*NP* $\subseteq$ *NP* und daher auch co-*NP* $=$ *NP* zur Folge. Im Umkehrschluss ergibt sich MEC $\notin$ *NP* aus der *NP* $\neq$ co-*NP*-Voraussetzung.

**qed.**

Wie angekündigt hat sich ergeben, dass MEC ein sehr hartes Problem ist. Wir komplettieren das Bild jetzt durch folgende

**Beobachtung 14.8** *Es gibt eine polynomielle NOTM, die mit Hilfe eines* SAT$^*$*-Orakels als Akzeptor von MEC auftritt.*

[1] Eine Klausel heiße *tautologisch*, wenn sie nicht falsifiziert werden kann (wie zum Beispiel die leere Klausel oder die Klausel $v_1 \vee \bar{v}_1$).

**Beweis** Die NOTM gehe auf einer Eingabe $(S, k)$ vor wie folgt:

1. Rate einen Schaltkreis $S'$ mit maximal $k$ Bausteinen.
2. Seien $f$ und $f'$ die von $S$ und $S'$ berechneten Booleschen Funktionen. Synthetisiere auf die offensichtliche Weise die Schaltkreise $S$ und $S'$ zu einem neuen Schaltkreis $S^{\oplus}$, welcher die Funktion
$$f \oplus f' := (f \wedge \bar{f}') \vee (\bar{f} \wedge f')$$
berechnet.
3. Befrage das SAT*-Orakel nach der Erfüllbarkeit von $S^{\oplus}$.
4. Akzeptiere genau dann, wenn $S^{\oplus}$ nicht erfüllbar ist.

Diese Vorgehensweise ist wegen

$$\begin{aligned} f \equiv f' &\Leftrightarrow f \oplus f' \equiv 0 \\ &\Leftrightarrow S^{\oplus} \text{ ist nicht erfüllbar} \end{aligned}$$

korrekt. Offensichtlich existiert genau dann, eine akzeptierende Rechnung zur Eingabe $(S, k)$, wenn $(S, k)$ zur Sprache MEC gehört. Die Laufzeit der skizzierten NOTM ist sicherlich polynomiell beschränkt. **qed.**

## 14.2 Definition und elementare Eigenschaften der Hierarchie

**Definition 14.9** *1. Sei $L \subseteq \Sigma^*$ eine Sprache.*

*(a) $P[L]$ bezeichne die Klasse aller Sprachen, die von einer polynomiellen DOTM mit Orakel $L$ akzeptiert werden können.*

*(b) $NP[L]$ bezeichne die Klasse aller Sprachen, die von einer polynomiellen NOTM mit Orakel $L$ akzeptiert werden können.*

*2. Sei $\mathcal{C}$ eine Klasse von Sprachen über Alphabet $\Sigma$.*

$$\begin{aligned} P[\mathcal{C}] &:= \bigcup_{L \in \mathcal{C}} P[L] \\ NP[\mathcal{C}] &:= \bigcup_{L \in \mathcal{C}} NP[L] \end{aligned}$$

*(Notationen $P^L, NP^L, P^{\mathcal{C}}, NP^{\mathcal{C}}$ anstelle von $P[L], NP[L], P[\mathcal{C}], NP[\mathcal{C}]$ sind ebenso gebräuchlich.)*

Zum Aufwärmen machen wir ein paar einfache Beobachtungen:

**Lemma 14.10** *1. $\mathcal{C} \subseteq P[\mathcal{C}] \subseteq NP[\mathcal{C}]$.*

2. $P[P] = P$ *und* $NP[P] = NP$.

3. *Falls* $L_0$ $\mathcal{C}$*-vollständig (unter polynomiellen Reduktionen) ist, dann gilt* $P[L_0] = P[\mathcal{C}]$ *und* $NP[L_0] = NP[\mathcal{C}]$.

**Beweis**

1. $P[\mathcal{C}] \subseteq NP[\mathcal{C}]$ ist klar, da eine DOTM als eine spezielle NOTM angesehen werden kann. Um die Inklusion $\mathcal{C} \subseteq P[\mathcal{C}]$ nachzuweisen, genügt es, $L \in P[L]$ für eine beliebige Sprache $L \subseteq \Sigma^*$ zu zeigen. Dies ist aber klar, da eine mit einem $L$-Orakel ausgestattete DOTM die Mitgliedschaft von $x$ in $L$ durch Befragung des $L$-Orakels nach $x$ unmittelbar testen kann.

2. Eine polynomielle DTM (oder gar NTM) kann die Arbeit eines $P$-Orakels auch gleich selber machen.

3. Wir beschränken uns auf die Aussage $P[L_0] = P[\mathcal{C}]$. Der Nachweis der Aussage $NP[L_0] = NP[\mathcal{C}]$ erfolgt analog.
   $P[L_0] \subseteq P[\mathcal{C}]$ ist die triviale Beweisrichtung. Zum Nachweis von $P[\mathcal{C}] \subseteq P[L_0]$ haben wir für eine beliebige Sprache $L \in \mathcal{C}$ die Inklusion $P[L] \subseteq P[L_0]$ zu zeigen. Hierzu nutzen wir die $\mathcal{C}$-Vollständigkeit von $L_0$ aus. Sei $f$ die polynomiell berechenbare Abbildung bei der polynomiellen Reduktion von $L$ nach $L_0$. Die an ein $L$-Orakel gerichtete Nachfrage nach einem Wort $x \in \Sigma^*$ kann dann durch eine an ein $L_0$-Orakel gerichtete Nachfrage nach dem Wort $f(x)$ ersetzt werden.

**qed.**

Aussage 1 von Lemma 14.10 besagt gewissermaßen, dass eine Klasse $\mathcal{C}$ durch Anwendung des „$P$-Operators“ zu einer Oberklasse $P[\mathcal{C}]$ „aufgeblasen“ wird. Die Anwendung des mächtigeren „$NP$-Operators“ bläst $\mathcal{C}$ zu einer (evtl.) noch größeren Klasse auf. In diesem Lichte bedeutet Aussage 2, dass der $P$-Operator auf $P$ stationär ist und der $NP$-Operator $P$ zu $NP$ aufbläst. Aussage 3 bedeutet, dass es keinen Unterschied macht, ob wir einen der Operatoren auf $\mathcal{C}$ oder eine $\mathcal{C}$-vollständige Sprache anwenden. Die folgende Definition der polynomiellen Hierarchie, welche sich zwischen $P$ bzw. $NP$ und $PSpace$ aufspannt, beruht auf iterativer Anwendung des $P$- und des $NP$-Operators.

**Definition 14.11** *Die Sprachklassen* $\Sigma_k$ *und* $\Delta_k$ *sind für* $k \geq 0$ *induktiv definiert wie folgt:*

$$\begin{aligned} \Sigma_0 &:= \Delta_0 := P \\ \Sigma_k &:= NP[\Sigma_{k-1}] \\ \Delta_k &:= P[\Sigma_{k-1}] \end{aligned}$$

*Die Sprachlassen* $\Pi_k$ *ergeben sich als Komplementärklassen zu den Sprachklassen* $\Sigma_k$*:*

$$\Pi_k := co\text{-}\Sigma_k \ .$$

*Die Klassen* $\Sigma_k, \Delta_k, \Pi_k$ *bilden (zusammen mit den zwischen ihnen geltenden Inklusionsbeziehungen) die sogenannte* polynomielle Hierarchie.

Die untersten drei Stufen dieser Hierarchie ergeben sich aus dieser Definition wie folgt:

**Stufe 0** $\Sigma_0 = \Delta_0 = P$ und $\Pi_0 = \text{co-}P = P$.

**Stufe 1** $\Sigma_1 = NP[P] = NP$, $\Delta_1 = P[P] = P$ und $\Pi_1 = \text{co-}NP$.

**Stufe 2** $\Sigma_2 = NP[NP]$, $\Delta_2 = P[NP]$ und $\Pi_2 = \text{co-}NP[NP]$.

Wir sind jetzt darauf vorbereitet, die Beobachtung 14.8 etwas gelehrsamer zu formulieren:

**Lemma 14.12** *MEC* $\in \Sigma_2$.

**Beweis** Da SAT$^*$ *NP*-vollständig ist, gilt $\Sigma_2 = NP[NP] = NP[\text{SAT}^*]$. Beobachtung 14.8 ist äquivalent zu MEC $\in NP[\text{SAT}^*]$. **qed.**

Die folgenden Lemmas formulieren elementare Eigenschaften der polynomiellem Hierarchie und sind darüberhinaus eine geeignete Aufwärmübung, die die Vertrautheit mit Definition 14.11 erhöhen sollte. Wir geben bei jedem Beweis nur die entscheidende Idee an.

Das folgende Lemma macht deutlich, dass die polynomielle Hierarchie sich nicht verändert, wenn wir den *P*- oder *NP*-Operator auf $\Pi_{k-1}$ anstelle von $\Sigma_{k-1}$ anwenden.

**Lemma 14.13** $\Sigma_k = NP[\Pi_{k-1}]$ *und* $\Delta_k = P[\Pi_{k-1}]$ *für alle* $k \geq 1$.

**Beweis** Nutze aus, dass ein $L$-Orakel gleichwertig zu einem $\bar{L}$-Orakel ist: statt nach $w \in L$ zu fragen, frage nach $w \in \bar{L}$ und negiere die Antwort. **qed.**

Das nächste Lemma besagt, dass die Klassen $\Delta_k$ abgeschlossen unter Komplementbildung sind.

**Lemma 14.14** $\Delta_k = \textit{co-}\Delta_k$ *für alle* $k \geq 0$.

**Beweis** Nutze aus, dass eine $L$ akzeptierende DOTM durch Vertauschen von akzeptierenden und nicht-akzeptierenden Zuständen in eine $\bar{L}$ akzeptierende DOTM transformiert wird. **qed.**

Das nun folgende Lemma deckt die Inklusionsbeziehungen auf:

**Lemma 14.15** $\Delta_k \subseteq \Sigma_k \cap \Pi_k$ *und* $\Sigma_k \cup \Pi_k \subseteq \Delta_{k+1}$ *für alle* $k \geq 0$.

**Beweis** Der Beweis erfolgt induktiv. Für $k = 0$ sind die Aussagen gültig. Für $k \geq 1$ ergibt sich zunächst

$$\Delta_k = P[\Sigma_{k-1}] \subseteq NP[\Sigma_{k-1}] = \Sigma_k \ .$$

Dies impliziert

$$\Delta_k = \text{co-}\Delta_k \subseteq \text{co-}\Sigma_k = \Pi_k$$

und daher auch $\Delta_k \subseteq \Sigma_k \cap \Pi_k$. Der Rest ergibt sich mit Hilfe der Lemmas 14.10 und 14.13:

$$\Sigma_k \subseteq P[\Sigma_k] = \Delta_{k+1} \text{ und } \Pi_k \subseteq P[\Pi_k] = \Delta_{k+1} \ .$$

**qed.**

Die Inklusionsbeziehungen in Lemma 14.15 sind in Abbildung 1 visualisiert. Es ergibt sich ausgehend von $P$ und $NP$ (auf den unteren Stufen) eine (voraussichtlich) unendliche Hierarchie von immer mächtigeren Klassen. Wir erhalten eine gemeinsame Oberklasse $PH$ durch die Definition

$$PH := \bigcup_{k \geq 0} \Sigma_k \ . \tag{1}$$

Wir werden später zeigen, dass (wie in Abbildung 1 bereits zu sehen) $PH$ in PSpace enthalten ist.

**Offene Probleme** Es ist nicht wirklich geklärt, ob $PH$ eine unendliche Hierarchie ist oder (das andere Extrem) vielleicht zu $NP$ oder gar $P$ kollabiert. Wir werden später sehen, dass $P = NP$ die scheinbar stärkere Aussage $P = PH$ impliziert. Auf ähnliche Weise würde $NP =$ co-$NP$, also Abgeschlossenheit von $NP$ unter Komplementbildung, $NP = PH$ implizieren. In Verbindung mit der $P \neq NP$-Vermutung bzw. der $NP \neq$ co-$NP$-Vermutung, gibt es den naheliegenden Verdacht, dass $PH$ nicht auf eine feste Stufe kollabiert, sondern dass jede neue Stufe der Hierarchie eine echt erweiterte Sprachklasse repräsentiert. In diesem Fall ergäbe sich eine unendliche Hierarchie. Eine vertiefte Debatte dieser Fragen werden wir führen können, wenn wir in einem späteren Abschnitt den Satz von Celia Wrathall kennengelernt haben, der die polynomielle Hierarchie zur prädikatenlogischen Beschreibungskomplexität in Beziehung setzt.

# 15 Der Satz von Celia Wrathall

Im Jahre 1977 fand Celia Wrathall eine prädikatenlogische Charaktersisierung der Klassen $\Sigma_k$ und $\Pi_k$, welche wir auf der $k$-ten Stufe der polynomiellen Hierarchie vorfinden. Wir erinnern uns, dass $\Sigma_k$ aus $\Sigma_{k-1}$ durch Anwendung des $NP$-Operators hervorgeht: $\Sigma_k = NP[\Sigma_{k-1}]$. Durch anschließenden Übergang zur Komplementärklasse erhalten wir $\Pi_k =$ co-$NP[\Sigma_{k-1}]$. Anschaulich können wir die Hintereinanderausführung des $NP$-Operators und des Übergangs zur Komplementärklasse als co-$NP$-Operator auffassen. Die Operatoren $NP$ und co-$NP$ sind berechnungstheoretischer Natur: die aktuelle Komplexitätsklasse wird erweitert durch Hinzufügen von „Rechenpower“. Celia Wrathall setzt implizit[2] an die Stelle der Operatoren $NP$ und co-$NP$ Operatoren prädikatenlogischer Natur. Wir notieren diese Operatoren im Folgenden durch $(\exists)_{pol}$ und $(\forall)_{pol}$. Es wird sich zeigen, dass die iterative Anwendung der neuen Operatoren die gleiche Hierarchie aufbaut wie die iterative Anwendung der ursprünglichen Operatoren. Der Aufbau der polynomiellen Hierarchie mit den Operatoren $(\exists)_{pol}$ uns $(\forall)_{pol}$ führt zu einer prädikatenlogischen Charakterisierung von $\Sigma_k$ und $\Pi_k$ vermöge alternierender Quantorenketten der Länge $k$. Der Satz von Celia Wrathall zeigt gewissermaßen wie sich die Berechnungskomplexität einer Sprache der polynomiellen Hierarchie in prädikatenlogische „Beschreibungskomplexität“ übersetzt (und umgekehrt).

[2] Wir weichen bei der Darstellung des Satzes von Celia Wrathall und seines Beweises von der Originalarbeit ab.

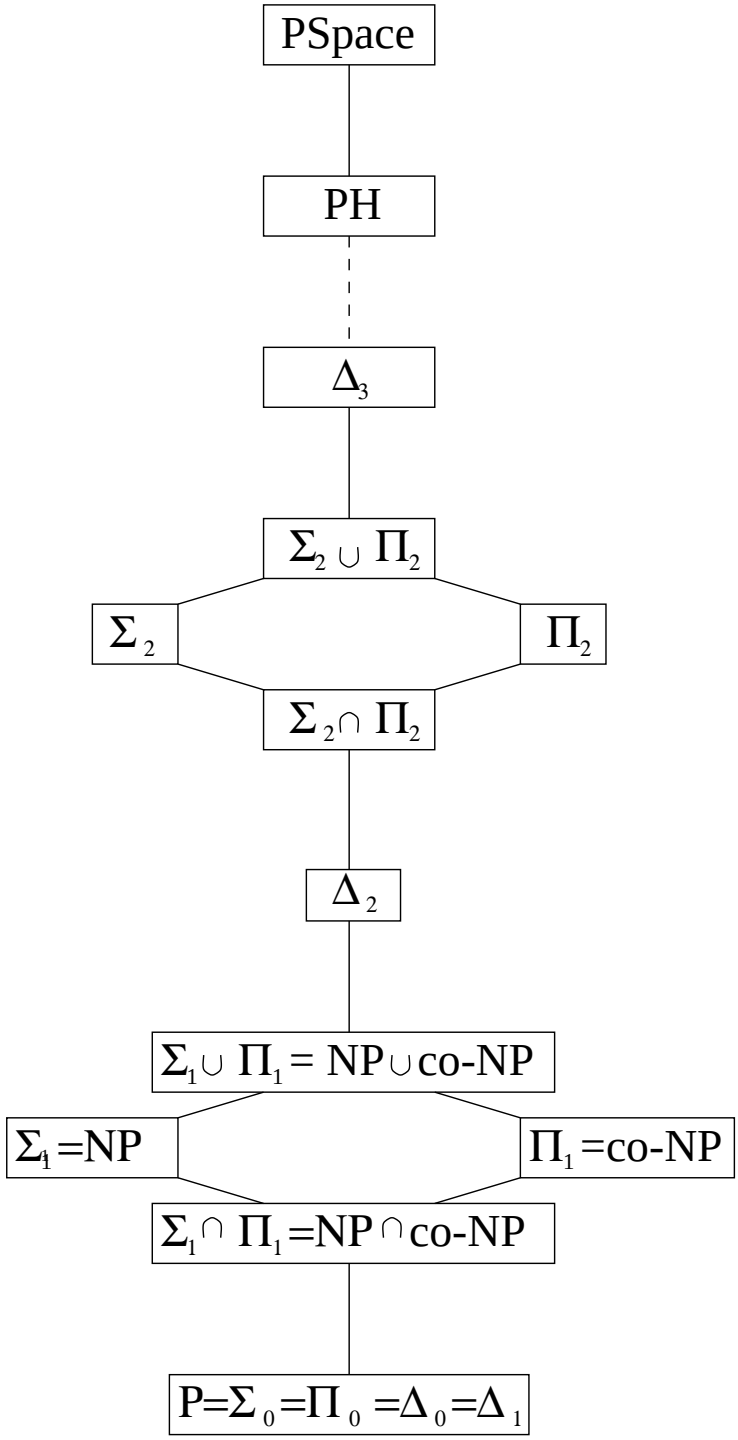


Abbildung 1: Die polynomielle Hierarchie von Stockmeyer.

Aus Gründen der Bequemlichkeit treffen wir nun noch ein paar notationelle Vereinbarungen:

1. Das Kürzel „pol“ steht im Folgenden stets für ein (geeignet gewähltes) Polynom $p$. Wir setzen oBdA stets voraus, dass $p$ platz- und zeitkonstruierbar ist.

2. Das Kürzel $(\exists y)_{pol}$ bzw. $(\forall y)_{pol}$ stehe für Quantifizierung (vom Typ „$\exists$“ bzw. „$\forall$“ über alle Wörter (gebildet über einem festen Alpabet) einer durch $\mathrm{pol}(|x|)$ beschränkten Länge, wobei der Bezugsstring $x$ stets aus dem Kontext hervorgeht.

3. Die Notation $\langle z_1, \ldots, z_k \rangle$ bezeichne eine fest ausgewählte „natürliche“ Kodierung des Tupels $(z_1, \ldots, z_k)$ durch *ein* Wort.

Eine mögliche natürliche Kodierung von $(z_1, \ldots, z_k)$ entsteht, wenn wir die Buchstaben von $z_1, \ldots, z_k$ verdoppeln und an den $k-1$ „Nahtstellen“ jeweils „01“ als Trennzeichen einschieben. Dann würde zum Beispiel gelten $\langle 101, \varepsilon, 00 \rangle = 11001101010000$. Hier, wie bei jeder anderen natürlichen Kodierung auch, ist die Länge des Codewortes $\langle z_1, \ldots, z_k \rangle$ linear in der Gesamtlänge von $z_1, z_2, \ldots, z_k$ beschränkt. Kodierung und Dekodierung kann jeweils in Linearzeit erfolgen.

## 15.1 Die prädikatenlogischen Operatoren

Nach der wortreichen Einleitung wird es nun Zeit konkret zu werden. Voilà, hier ist die genaue Definition der geheimnisvollen neuen Operatoren:

**Definition 15.1** *Sei $\mathcal{C}$ eine Klasse von Sprachen (über einem festen Alphabet).*

1. *$(\exists)_{pol}[\mathcal{C}]$ sei die Klasse aller Sprachen $L$, die sich für ein $L_0 \in \mathcal{C}$ in der Form*

$$L = \{x : (\exists y)_{pol} \langle y, x \rangle \in L_0\} \tag{2}$$

   *schreiben lassen.*

2. *$(\forall)_{pol}[\mathcal{C}]$ sei die Klasse aller Sprachen $L$, die sich für ein $L_0 \in \mathcal{C}$ in der Form*

$$L = \{x : (\forall y)_{pol} \langle y, x \rangle \in L_0\} \tag{3}$$

   *schreiben lassen.*

Da unsere Schreibkonventionen zwar bequem aber zugegebenermaßen auch etwas schlampig sind, geben wir hier (zum letzten Mal) zusätzlich die ausführliche Schreibweise an (um Missverständnissen vorzubeugen). Gleichung (2) bedeutet: Es existiert ein Polynom $p$, so dass die Sprache $L$ genau diejenigen Wörter $x$ enthält, zu denen ein Wort $y$ der Länge höchstens $p(|x|)$ existiert, so dass der Kodierungsstring für das Paar $(y, x)$ in $L_0$ zu liegen kommt. Gleichung (3) bedeutet: Es existiert ein Polynom $p$, so dass die Sprache $L$ genau diejenigen Wörter $x$ enthält, welche die Eigenschaft haben, dass für alle Wörter $y$ der Länge höchstens

$p(|x|)$ der Kodierunsstring für das das Paar $(y, x)$ in $L_0$ zu liegen kommt. (Uff, es lebe die Schlampigkeit!)

Für einen Quantor $Q \in \{\exists, \forall\}$ bezeichne seine Negation $\bar{Q}$ den jeweils anderen „dualen" Quantor:

$$\bar{Q} := \begin{cases} \forall, & \text{falls } Q = \exists, \\ \exists, & \text{falls } Q = \forall. \end{cases} \tag{4}$$

Wir werden im Folgenden häufig mit alternierenden Quantorenketten arbeiten. Um diese bequem zu notieren, definieren wir

$$Q_k := \begin{cases} \exists, & \text{falls } k \text{ ungerade ist}, \\ \forall, & \text{falls } k \text{ gerade ist}. \end{cases} \tag{5}$$

Trivialerweise gilt dann

$$\bar{Q}_k = \begin{cases} \forall, & \text{falls } k \text{ ungerade ist}, \\ \exists, & \text{falls } k \text{ gerade ist}. \end{cases} \tag{6}$$

Eine alternierende Quantorenkette der Länge $k$ hat dann die Form

$$(\exists)_{pol}(\forall)_{pol}(\exists)_{pol} \cdots (Q_k)_{pol} \ ,$$

wenn sie mit $(\exists)_{pol}$ beginnt. Beginnt sie mit $(\forall)_{pol}$ hat sie die Form

$$(\forall)_{pol}(\exists)_{pol}(\forall)_{pol} \cdots (\bar{Q}_k)_{pol} \ .$$

Die analogen Schreibweisen benutzen wir auch für $(\exists y)_{pol}$ und $(\forall y)_{pol}$.

Es folgen ein paar Aufwärmübungen zu den neuen Definitionen und Schreibweisen.

**Lemma 15.2** *Beim Übergang zur Komplementärklasse gelten die folgenden Regeln:*

$$co\text{-}(\exists)_{pol}(\forall)_{pol}(\exists)_{pol} \cdots (Q_k)_{pol}[\mathcal{C}] = (\forall)_{pol}(\exists)_{pol}(\forall)_{pol} \cdots (\bar{Q}_k)_{pol}[co\text{-}\mathcal{C}] \ . \tag{7}$$

$$co\text{-}(\forall)_{pol}(\exists)_{pol}(\forall)_{pol} \cdots (\bar{Q}_k)_{pol}[\mathcal{C}] = (\exists)_{pol}(\forall)_{pol}(\exists)_{pol} \cdots (Q_k)_{pol}[co\text{-}\mathcal{C}] \ . \tag{8}$$

**Beweis** Wir beschränken uns auf den Nachweis von (7). Der Nachweis von (8) lässt sich völlig analog führen.
$k$-fache Anwendung von Definition 15.1 führt zu folgender Einsicht. Zu einer Sprache $L \in (\exists)_{pol}(\forall)_{pol}(\exists)_{pol} \cdots (Q_k)_{pol}[\mathcal{C}]$ gibt es eine Sprache $L_0 \in \mathcal{C}$, so dass $L$ gegeben ist durch

$$L = \{x : (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_k y_k)_{pol}\langle y_1, \ldots, y_k, x\rangle \in L_0\} \ . \tag{9}$$

Die Komplementärklasse co-$\mathcal{C}$ enthält anstelle von $L$ die Sprache $\bar{L}$, deren Definitionsbedingung sich durch logische Negation der Definitionsbedingung von $L$ ergibt:

$$\bar{L} = \{x : (\forall y_1)_{pol}(\exists y_2)_{pol}(\forall y_3)_{pol} \cdots (\bar{Q} y_k)_{pol}\langle y_1, \ldots, y_k, x\rangle \in \bar{L}_0 \ . \tag{10}$$

Hierbei haben wir die de Morgan'schen Regeln angewendet, welche dazu führen, dass jeder Quantor durch sein duales Gegenstück ersetzt und die atomore Bedingung $\langle y_1, \ldots, y_k, x\rangle \in L_0$ am Ende der Quantorenkette zu $\langle y_1, \ldots, y_k, x\rangle \in \bar{L}_0$ logisch negiert wird. Durch $k$-fache Anwendung von Definition 15.1 erkennen wir jetzt, dass die Bedingung (10) gerade Sprachen aus $(\forall)_{pol}(\exists)_{pol}(\forall)_{pol} \cdots (\bar{Q}_k)_{pol}[\text{co-}\mathcal{C}]$ charakterisiert. **qed.**

Wegen co-$P = P$ ergibt sich unmittelbar die

**Folgerung 15.3**

$$\begin{aligned} co\text{-}(\exists)_{pol}(\forall)_{pol}(\exists)_{pol}\cdots(Q_k)_{pol}[P] &= (\forall)_{pol}(\exists)_{pol}(\forall)_{pol}\cdots(\bar{Q}_k)_{pol}[P] \ . \\ co\text{-}(\forall)_{pol}(\exists)_{pol}(\forall)_{pol}\cdots(\bar{Q}_k)_{pol}[P] &= (\exists)_{pol}(\forall)_{pol}(\exists)_{pol}\cdots(Q_k)_{pol}[P] \ . \end{aligned}$$

Ähnlich leicht erhalten wir die

**Folgerung 15.4** *$NP = (\exists)_{pol}[P]$ und co-$NP = (\forall)_{pol}[P]$.*

**Beweis** Die Aussage $NP = (\exists)_{pol}[P]$ entpuppt sich mit etwas Nachdenken als unsere gute alte Charakterisierung von $NP$ mit Hilfe von polynomiell verifizierbaren Relationen (Stichwort: Rate- und Verifikationsprinzip). Die zweite Aussage ergibt sich dann mit Hilfe von Folgerung 15.3: co-$NP$ = co-$(\exists)_{pol}[P] = (\forall)_{pol}[P]$. **qed.**

Wir stellen uns als nächstes die Frage, ob die Operatoren $(\exists)_{pol}$ und $(\forall)_{pol}$ zu einer (evtl. unechten) Erweiterung einer Sprachklasse führen. Vorsicht ist insofern geboten, als man künstliche Sprachklassen $\mathcal{C}$ **angeben kann**, für welche die Inklusionsbeziehung $\mathcal{C} \subseteq (\exists)_{pol}[\mathcal{C}]$ **Übg.** bzw. $\mathcal{C} \subseteq (\forall)_{pol}[\mathcal{C}]$ *nicht* gilt. Andererseits beweisen wir folgendes

**Lemma 15.5** *$\mathcal{C}$ besitze die die folgende Abschluss-Eigenschaft. Falls $L \in \mathcal{C}$, dann gehöre auch die Sprache*

$$L_\epsilon := \{\langle \epsilon, x\rangle : x \in L\}$$

*zu $\mathcal{C}$. (Hierbei bezeichne $\epsilon$ wie üblich das leere Wort.) Unter dieser Voraussetzung gilt $\mathcal{C} \subseteq (\exists)_{pol}[\mathcal{C}]$ und $\mathcal{C} \subseteq (\forall)_{pol}[\mathcal{C}]$.*

**Beweis** Sei $L \in \mathcal{C}$ und somit auch $L_\epsilon \in \mathcal{C}$. Um $\mathcal{C} \subseteq (\exists)_{pol}[\mathcal{C}]$ einzusehen, genügt es $L \in (\exists)_{pol}[\mathcal{C}]$ einzusehen. Dies ist aber klar, da wir $L$ in folgender Form schreiben können

$$L = \{x : (\exists y)_{pol}\langle y, x\rangle \in L_\epsilon\} \ .$$

Das geeignete Polynom, das sich hinter dem Index „pol“ von $(\exists y)_{pol}$ versteckt, kann in diesem Fall als das Nullpolynom $p \equiv 0$ gewählt werden. In der Tat existiert ja für die Wörter $x$ von $L$ (und nur für diese) ein Wort $y$ einer Länge 0 (nämlich das leere Wort $y = \epsilon$) mit $\langle y, x\rangle \in L_\epsilon$. Der Nachweis von $\mathcal{C} \subseteq (\forall)_{pol}[\mathcal{C}]$ geschieht völlig analog (oder durch Dualisierung). **qed.**

Es ist **leicht einzusehen**, dass alle Sprachklassen der polynomiellen Hierarchie (wie über- **Übg.** haupt alle „vernünftig definierten“ Komplexitätsklassen) die in Lemma 15.5 geforderte Abschluss-Eigenschaft besitzen. Für $\Sigma_k$ und $\Pi_k$ zeigen wir aber darüberhinaus das

**Lemma 15.6** *Für alle $k \geq 1$ gilt $\Sigma_k = (\exists)_{pol}[\Sigma_k]$ und $\Pi_k = (\forall)_{pol}[\Pi_k]$.*

**Beweis** Wir zeigen zunächst $\Sigma_k = (\exists)_{pol}[\Sigma_k]$. $\Sigma_k \subseteq (\exists)_{pol}[\Sigma_k]$ ergibt sich aus Lemma 15.5. Zum Nachweis von $(\exists)_{pol}[\Sigma_k] \subseteq \Sigma_k$ sei $L \in (\exists)_{pol}[\Sigma_k]$ für ein $L_0 \in \Sigma_k$ gegeben durch

$$L = \{x : (\exists y)_{pol}\langle y, x\rangle \in L_0\} \ .$$

Sei $M_0$ ein Akzeptor von $L_0 \in \Sigma_k$. Wegen $k \geq 1$ ist $M_0$ eine polynomielle NOTM mit einem Orakel für eine Sprache $L_0' \in \Sigma_{k-1}$. (Dies schließt in einer etwas verklausulierten Form auch den Fall $k = 1$ ein.) Folgende NOTM $M$, versehen mit einem Orakel für $L_0' \in \Sigma_{k-1}$, ist ein polynomieller Akzeptor von $L$:

1. Rate ein Wort $y$ der Länge $\text{pol}(|x|)$.
2. Benutze die NOTM $M_0$ mit dem $L_0'$-Orakel, um $\langle y, x\rangle \in L_0$ zu verifizieren (falls möglich).

Man überlegt sich leicht, dass $M$ auf einer Eingabe $x$ genau dann eine akzeptierende Rechnung besitzt, wenn $x \in L$.
Mit unserem Hintergrundwissen ergibt sich nun $\Pi_k = (\forall)_{pol}[\Pi_k]$ leicht durch Dualisierung:

$$\Pi_k = \text{co-}\Sigma_k = \text{co-}(\exists)_{pol}[\Sigma_k] = (\forall)_{pol}[\text{co-}\Sigma_k] = (\forall)_{pol}[\Pi_k] \ .$$

**qed.**

Im Beweis von $\Sigma_k = (\exists)_{pol}[\Sigma_k]$ haben wir (salopp gesprochen) ausgenutzt, dass eine NOTM ihren Nichtdeterminismus dazu verwenden kann, den $(\exists)_{pol}$-Operator (durch Raten eines geeigneten Wortes $y$) überflüssig zu machen.

Aus beweistechnischen Gründen definieren wir jetzt die polynomielle Hierarchie ein zweites Mal (wie sich später herausstellen wird), wobei diesmal die neuen prädikatenlogischen Operatoren zum Einsatz kommen:

**Definition 15.7** *Die Sprachklassen $\Sigma_k'$ und $\Pi_k'$ sind für $k \geq 0$ induktiv definiert wie folgt:*

$$\begin{aligned} \Sigma_0' &:= \Pi_0' := P \ . \\ \Sigma_k' &:= (\exists)_{pol}[\Pi_{k-1}'] \ . \\ \Pi_k' &:= (\forall)_{pol}[\Sigma_{k-1}'] \ . \end{aligned}$$

Der Satz von Celia Wrathall besagt, dass $\Sigma_k' = \Sigma_k$ und $\Pi_k' = \Pi_k$, d.h., die „neue“ Hierachie ist die „alte“ Hierarchie in verkleideter Form. Der Beweis dieser Aussage ist nichttrivial und wird in Abschnitt 15.2 geführt.

Iterierte Anwendung von Definition 15.7 liefert sofort eine Charakterierung der Klassen $\Sigma_k'$ und $\Pi_k'$ vermöge alternierender Quantorenketten:

**Lemma 15.8** *Für alle $k \geq 1$ gilt:*

$$\begin{aligned} L \in \Sigma_k' &\Leftrightarrow \exists L_0 \in P : L = \{x : (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_k y_k)_{pol}\langle y_1, \ldots, y_k, x\rangle \in L_0\} \ . \\ L \in \Pi_k' &\Leftrightarrow \exists L_0 \in P : L = \{x : (\forall y_1)_{pol}(\exists y_2)_{pol}(\forall y_3)_{pol} \cdots (\bar{Q} y_k)_{pol}\langle y_1, \ldots, y_k, x\rangle \in L_0\} \ . \end{aligned}$$

Aus der Charakterisierung der Klassen $\Sigma_k', \Pi_k'$ mit alternierenden Quantorenketten lassen sich unmittelbar eine paar Schlussfolgerungen ziehen.

**Folgerung 15.9** *1. Für alle $k \geq 0$ gilt $\Pi_k' = \text{co-}\Sigma_k'$.*

2. *Für alle* $k \geq 1$ *gilt* $\Sigma'_k = (\exists)_{pol}[\Sigma'_k]$.

3. *Für alle* $k \geq 1$ *gilt* $\Pi'_k = (\forall)_{pol}[\Pi'_k]$.

4. *Für alle* $k \geq 0$ *gilt* $\Sigma'_k \subseteq \Sigma'_{k+1} \cap \Pi'_{k+1}$ *und* $\Pi'_k \subseteq \Pi'_{k+1} \cap \Sigma'_{k+1}$.

**Beweis** Wir geben jeweils nur die entscheidende Idee an. (Die technischen Details sind leicht auszufüllen.)

1. Kombiniere Lemma 15.8 mit Lemma 15.2.

2. Die Grundidee ist, dass $(\exists)_{pol}(\exists)_{pol}$ gleichwertig zu $(\exists)_{pol}$ ist, da sich zwei aufeinanderfolgende Quantoren des gleichen Typs miteinander verschmelzen lassen.

3. Das geschilderte Verschmelzungsargument gilt natürlich auch für $(\forall)_{pol}$-Quantoren. Alternativ kann $\Pi'_k = (\forall)_{pol}[\Pi'_k]$ auch leicht durch Dualisierung gezeigt werden, wobei dann die bereits bewiesenen Aussagen $\Pi'_k = \text{co-}\Sigma'_k$ und $\Sigma'_k = (\exists)_{pol}[\Sigma'_k]$ benutzt werden.

4. Die „Beschreibungskraft" einer Quantorenkette kann nicht abnehmen, wenn wir sie um einen neuen Quantor verlängern.

**qed.**

Die Ausführung der technischen Details zu dem „Verschmelzungsargument" sind Bestandteil empfehlen wir als **Übung**. Man kann sich dabei an der technischen Beweisführung des folgenden Lemmas orientieren. **Übg.**

**Lemma 15.10** *Für jedes* $k \geq 0$ *sind die Klassen* $\Sigma'_k$ *und* $\Pi'_k$ *abgeschlossen unter Vereinigung und Durchschnitt von Sprachen.*

**Beweis** Für $k = 0$ ist die Aussage offensichtlich. Sei $k \geq 1$. Seien $L', L'' \in \Sigma'_k$. Gemäß Lemma 15.8 existieren dann Sprachen $L'_0, L''_0 \in P$, so dass

$$\begin{aligned} L' &= \{x : (\exists y'_1)_{pol}(\forall y'_2)_{pol}(\exists y'_3)_{pol} \cdots (Q_k y'_k)_{pol} \langle y'_1, \ldots, y'_k, x\rangle \in L'_0\} \ . \\ L'' &= \{x : (\exists y''_1)_{pol}(\forall y''_2)_{pol}(\exists y''_3)_{pol} \cdots (Q_k y''_k)_{pol} \langle y''_1, \ldots, y''_k, x\rangle \in L''_0\} \ . \end{aligned}$$

Hieraus ergibt sich

$$\begin{aligned} L' \cup L'' &= \{x : (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_k y_k)_{pol} \, (y_1 = \langle y'_1, y''_1\rangle \wedge \cdots \wedge y_k = \langle y'_k, y''_k\rangle) \\ &\quad \wedge \ (\text{Längenbedingung}) \ \wedge (\langle y'_1, \ldots, y'_k, x\rangle \in L'_0 \vee \langle y''_1, \ldots, y''_k, x\rangle \in L''_0)\} \ . \\ L' \cap L'' &= \{x : (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_k y_k)_{pol} \, (y_1 = \langle y'_1, y''_1\rangle \wedge \cdots \wedge (y_k = \langle y'_k, y''_k\rangle) \\ &\quad \wedge \ (\text{Längenbedingung}) \ \wedge (\langle y'_1, \ldots, y'_k, x\rangle \in L'_0 \wedge \langle y''_1, \ldots, y''_k, x\rangle \in L''_0)\} \ . \end{aligned}$$

Hierbei steht (Längenbedingung) für eine Bedingung, welche kontrolliert, dass $y'_1, \ldots, y'_k$ und $y''_1, \ldots, y''_k$ so längenbeschränkt sind wie es durch die Definition von $L'$ bzw. $L''$ vorgeschrieben

ist. Wenn also $p'_i$ bzw. $p''_i$ die konkreten Polynome sind, welche die Längen von $y'_i$ bzw. $y''_i$ nach oben beschränken, dann ist (Längenbedingung) die logische Konjunktion der Bedingungen

$$|y'_i| \leq p'_i(|x|) \wedge |y''_i| \leq p''_i(|x|)$$

für $i = 1, \ldots, k$. Man beachte, dass sowohl die Längenbedingung als auch die logische Konjunktion der Bedingungen $y_i = \langle y'_i, y''_i \rangle$ für $i = 1, \ldots, k$ in Polynomialzeit getestet werden kann.[3] Eine Inspektion der obigen Definitionsbedingungen für $L' \cup L''$ bzw. $L' \cap L''$ verbunden mit einer erneuten Anwendung von Lemma 15.8 liefert $L' \cup L'', L' \cap L'' \in \Sigma'_k$.
Die entsprechenden Abschlusseigenschaften von $\Pi'_k$ **ergeben sich leicht** durch Dualisierung. **Übg.**

**qed.**

## 15.2 Der Satz von Celia Wrathall und sein Beweis

Im vorangegangenen Abschnitt haben wir genug Vorarbeit geleistet, um jetzt dem Hauptresultat zu Leibe zu rücken:

**Satz 15.11 (Wrathall, 1977)** *Für alle $k \geq 0$ gilt $\Sigma'_k = \Sigma_k$ und $\Pi'_k = \Pi_k$.*

**Beweis** Für $k = 0$ ist die Behauptung richtig, da auf Stufe 0 alle beteiligten Sprachklassen mit $P$ zusammenfallen. Wir können daher für ein beliebiges aber festes $k \geq 1$ induktiv $\Sigma'_{k-1} = \Sigma_{k-1}$ und $\Pi'_{k-1} = \Pi_{k-1}$ voraussetzen. Aus $\Sigma'_k = \Sigma_k$ ergibt sich $\Pi'_k = \Pi_k$ durch Dualisierung:

$$\Pi'_k = \text{co-co-}\Pi'_k = \text{co-}\Sigma'_k = \text{co-}\Sigma_k = \Pi_k \ .$$

Es genügt also der induktive Nachweis von $\Sigma'_k = \Sigma_k$.
Die Inklusion $\Sigma'_k \subseteq \Sigma_k$ ergibt sich wie folgt:

$$\Sigma'_k = (\exists)_{pol}[\Pi'_{k-1}] = (\exists)_{pol}[\Pi_{k-1}] \subseteq (\exists)_{pol}[\Sigma_k] = \Sigma_k \ .$$

Hierbei haben wir nacheinander die Definition von $\Sigma'_k$, die Induktionsvoraussetzung, die Inklusion $\Pi_{k-1} \subseteq \Sigma_k$ (die nach Anwendung des $(\exists)_{pol}$-Operators gültig bleibt) und schließlich Lemma 15.6 angewendet.
Zum Nachweis der Inklusion $\Sigma_k \subseteq \Sigma'_k$ werden wir härter arbeiten müssen. Sei $L \in \Sigma_k = NP[\Sigma_{k-1}]$. Sei weiter $M$ die polynomielle NOTM, die in Kooperation mit einem Orakel für $L' \in \Sigma_{k-1}$ als Akzeptor von $L$ auftritt. $M$ darf als nichtdeterministische Maschine „raten“ und als Orakel-Maschine zusätzlich pol($|x|$)-oft ihr $L'$-Orakel befragen. Zum Nachweis von $L \in \Sigma'_k = (\exists)_{pol}[\Pi'_{k-1}]$ müssen wir die Fähigkeiten von $M$ irgendwie mit Hilfe des $(\exists)_{pol}$-Operators nachbilden. Die Korrespondenz von „Raten“ und $(\exists)_{pol}$-Quantifizierung schreckt uns nicht wirklich, da wir diese vom Rate- und Verifikationsprinzip her gewohnt sind. Bleibt aber das Problem, auch die Kommunikation zwischen $M$ und dem $L'$-Orakel mit Hilfe des $(\exists)_{pol}$-Quantors nachzubilden. Wir verwenden dabei die folgende

[3] Beim effizienten Testen der Längenbedingung nutzen wir die Platzkonstruierbarkeit der betreffenden Polynome aus.

**Zentrale Beweisstrategie** Antizipiere die Ratebits von $M$ *und* die gesamte Kommunikation zwischen $M$ und dem $L'$-Orakel durch einen einzigen (raffiniert entworfenen) String polynomiell beschränkter Länge und wende die $(\exists)_{pol}$-Quantifizierung auf diesen String an.

Es folgt die technische Umsetzung. $M$ stoppe nach höchstens $T = \mathrm{pol}(|x|)$ Rechenschritten und habe oBdA pro Schritt nur zwei Handlungsalternativen. Die Rechnung von $M$ hängt von folgenden Faktoren ab:

- den nichtdeterministischen Entscheidungen repräsentiert durch einen Ratestring $y \in \{0,1\}^T$,
- den JA/NEIN-Antworten des $L'$-Orakels.

Für $r, s \leq T$ seien $u, v$ Strings von der Form

$$u = \langle u_1, \ldots, u_r \rangle \text{ und } v = \langle v_1, \ldots, v_s \rangle \ .$$

Der String $\langle y, u, v, x \rangle$ heiße *konsistentes Rechenprotokoll*, falls folgendes gilt:

- Wenn $M$ auf Eingabe $x$ gemäß Ratestring $y$ rechnet und wenn alle Orakelanfragen nach einem Wort aus $U := \{u_1, \ldots, u_r\}$ mit JA und alle Orakelanfragen nach einem Wort aus $V := \{v_1, \ldots, v_s\}$ mit NEIN beantwortet werden, dann stellt $M$ nur Anfragen nach Wörtern aus $U \cup V$ und akzeptiert nach höchstens $T$ Schritten.

Beachte, dass wir in dieser Definition *nicht* verlangen, dass die Antworten korrekt sind. Stattdessen sind die (evtl. falschen) Antworten durch die Strings $u, v$ vorgegeben.[4] Die Sprache

$$L_0 := \{\langle y, u, v, x \rangle : \langle y, u, v, x \rangle \text{ ist ein konsistentes Rechenprotokoll}\}$$

gehört daher zu $P$.[5] Wir können nämlich auf Eingaben der Form $\langle y, u, v, x \rangle$ eine effiziente Simulation von $M$ auf $x$ starten[6], welche die konsistenten Rechenprotokolle (und nur diese) akzeptiert. Wir betrachten zwei Fälle:

**Fall 1** $M$ (mit Ratestring $y$) stelle während der ersten $T$ simulierten Schritte nur Orakelanfragen aus $U \cup V$.
Dann können wir ohne Probleme $T$ Schritte von $M$ simulieren, da die Antwort auf Orakelanfragen effizient aus dem String $u$ bzw. $v$ abgelesen werden kann. Wenn $M$ nach spätestens $T$ Schritten (akzeptierend oder verwerfend) stoppt, dann stoppe die Simulation mit dem gleichen Ergebnis. Wenn $M$ nach $T$ Schritten nicht stoppt[7], dann stoppe die Simulation nach $T$ simulierten Schritten verwerfend.[8]

[4] Dies ist eine subtile Angelegenheit: unsere Analyse der NOTM $M$ beschäftigt sich mit dem Verhalten von $M$, für den Fall, dass wir sie vom Orakel abkoppeln und ihr auf andere Weise Antworten verabreichen. In mathematischen Beweisen ist alles erlaubt! Warten wir mal ab, wozu solche dubiosen Manöver gut sind.

[5] Dazu diente das Manöver! Der Nichtdeterminismus wird mit Hilfe des Strings $y$ eliminiert; das Orakel wird mit Hilfe der Strings $u, v$ überflüssig gemacht; bleibt eine deterministische polynomielle Rechnung.

[6] Eingaben, die schon syntaktisch von der Form $\langle y, u, v, x \rangle$ abweichen, können sofort verworfen werden.

[7] was man wegen der $M$ zugespielten falschen Antworten nicht ausschließen kann

[8] An dieser Stelle verwenden wir die Zeitkonstruierbarkeit von $T = \mathrm{pol}(|x|)$.

**Fall 2** $M$ (mit Ratestring $y$) stelle während der ersten $T$ simulierten Schritte eine Orakelanfrage außerhalb $U \cup V$.
Bei der ersten Anfrage dieser Art stoppe die Simulation verwerfend.

Aus der Definition der konsistenten Rechenprotokolle ergibt sich direkt, dass unsere Simulation diese und nur diese akzeptiert. Somit gilt $L_0 \in P$.
Wir haben mit dem Nachweis von $L_0 \in P$ bereits einen Teilsieg errungen. Auf Dauer können wir uns aber nicht davor drücken, die Korrektheit von Antworten auf Orakelanfragen zu kontrollieren. Dieser Kontrolle dienen die folgenden beiden Sprachen:

$$\begin{aligned} L_1 &:= \{\langle y, u, v, x\rangle : u_1, \ldots, u_r \in L'\} \ . \\ L_2 &:= \{\langle y, u, v, x\rangle : v_1, \ldots, v_s \in \bar{L}'\} \ . \end{aligned}$$

Wegen $L' \in \Sigma_{k-1}$ und $\bar{L}' \in \text{co-}\Sigma_{k-1} = \Pi_{k-1}$ ist unter Verwendung der Induktionsvoraussetzung und des Lemmas 15.5 **leicht zu sehen**, dass folgende Aussagen gelten: **Übg.**

$$\begin{aligned} L_1 \in \Sigma_{k-1} &= \Sigma'_{k-1} \subseteq \Sigma'_k \ . \\ L_2 \in \Pi_{k-1} &= \Pi'_{k-1} \subseteq \Sigma'_k \ . \end{aligned}$$

Da $L_0 \in P \subseteq \Sigma'_k$ und $\Sigma'_k$ abgeschlossen unter Durchschnittsbildung ist, gilt dann auch $L_0 \cap L_1 \cap L_2 \in \Sigma'_k$. Weiterhin sind $L_0, L_1, L_2$ gerade so entworfen, dass

$$x \in L \Leftrightarrow (\exists w)_{pol} w = \langle y, u, v\rangle \wedge \langle y, u, v, x\rangle \in L_0 \cap L_1 \cap L_2 \ .$$

Hieraus ergibt sich leicht $L \in (\exists)_{pol}[\Sigma'_k]$ und wegen $(\exists)_{pol}[\Sigma'_k] = \Sigma'_k$ (gemäß Folgerung 15.9) folgt schließlich $L \in \Sigma'_k$. Da $L$ eine beliebige Sprache aus $\Sigma_k$ ist, ergibt sich die angestrebte Inklusion $\Sigma_k \subseteq \Sigma'_k$. **qed.**

In Verbindung mit Lemma 15.8 erhalten wir unmittelbar die

**Folgerung 15.12** *Für alle $k \geq 1$ gilt:*

$$\begin{aligned} L \in \Sigma_k &\Leftrightarrow \exists L_0 \in P : L = \{x : (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_k y_k)_{pol} \langle y_1, \ldots, y_k, x\rangle \in L_0\} \\ L \in \Pi_k &\Leftrightarrow \exists L_0 \in P : L = \{x : (\forall y_1)_{pol}(\exists y_2)_{pol}(\forall y_3)_{pol} \cdots (\bar{Q} y_k)_{pol} \langle y_1, \ldots, y_k, x\rangle \in L_0\} \end{aligned}$$

Die „Rechenkraft“ einer zu $\Sigma_k$ korrespondierenden polynomiellen NOTM (mit einem Orakel aus $\Sigma_{k-1}$) entspricht also exakt der „Beschreibungskraft“ einer alternierenden mit $(\exists)_{pol}$ beginnenden Quantorenkette der Länge $k$, gefolgt von einer in Polynomialzeit überprüfbaren Aussage. Die analoge Entsprechung finden wir für die Klasse $\Pi_k$ und alternierende mit $(\forall)_{pol}$ beginnende Quantorenketten der Länge $k$. Es hat sich somit eine erstaunliche Querbeziehung zwischen Berechnungskomplexität und prädikatenlogischer Beschreibungskomplexität ergeben.

## 15.3 Anwendungen des Satzes von Celia Wrathall

Für die harte Arbeit, die zum Beweis des Satzes von Celia Wrathall nötig war, werden wir in diesem Abschnitt entlohnt. Einige elegante Anwendungen dieses Satzes werden uns nun zufallen wie reife Früchte. Eine erste Anwendung besteht darin, die Einordnung einer Sprache in die polynomielle Hierarchie nicht durch „Programmierung“ einer geeigneten OTM vorzunehmen, sondern durch Angabe einer geeigneten „Definition“ der Sprache. Die Länge der alternierenden Quantorenkette in der Definitionsbedingung wird uns dann verraten, zu welcher Stufe der polynomiellen Hierchie die Sprache gehört. Eine zweite Anwendung ist die Analyse, unter welchen Bedingungen die polynomielle Hierachie auf eine endliche Stufe kollabiert. Eine dritte Anwendung besteht in der Angabe von $\Sigma_k$- oder $\Pi_k$-vollständigen Problemen und im Nachweis der Vollständigkeit.

### 15.3.1 Anwendung 1: Deskriptiver Nachweis der Mitgliedschaft in PH

Ein „algorithmischer“ Nachweis von $L \in PH$ besteht in der Angabe einer passenden OTM. Ein „deskriptiver“ Nachweis von $L \in PH$ macht sich Folgerung 15.12 zunutze und beschreibt $L$ mit einer passenden prädikatenlogischen Formel. Wir illustrieren dies an einem

**Beispiel 15.13** *Wir betrachten erneut die Sprache MEC (Minimum Equivalent Circuit). Wir haben früher bereits einen algorithmischen Nachweis von $MEC \in \Sigma_2$ geführt, und zwar durch Angabe einer polynomiellen NOTM, die mit Hilfe eines* SAT$^*$*-Orakels als Akzeptor von MEC auftrat. Es folgt nun ein deskriptiver Nachweis für die gleiche Aussage:*

$$MEC = \{\langle S, k\rangle : (\exists S')_{pol}(\forall a)_{pol}\langle S', a, S, k\rangle \in MEC_0\} \ ,$$

*wobei $MEC_0$ die Sprache aller Strings der Form $\langle S', a, S, k\rangle$ sei, so dass zusätzlich die folgenden Bedingungen gelten:*

1. *$S$ repräsentiert einen Booleschen Schaltkreis. Die Anzahl der Eingangsknoten in $S$ sei mit $n$ bezeichnet.*
2. *$k$ repräsentiert eine nicht-negative ganze Zahl.*
3. *$S'$ repräsentiert einen Booleschen Schaltkreis mit $n$ Eingangsknoten und höchstens $k$ Bausteinen.*
4. *$a \in \{0,1\}^n$ und die Schaltkreise $S$ und $S'$ berechnen auf Eingabe $a$ das gleiche Ausgabebit.*

*Offensichtlich gilt $MEC_0 \in P$. Aus Folgerung 15.12 ergibt sich unmittelbar $MEC \in \Sigma_2$.*

### 15.3.2 Anwendung 2: Hinreichende Bedingungen für einen Kollaps von PH

Es wird vermutet, dass die polynomielle Hierarchie aus unendlich vielen voneinander verschiedenen Stufen besteht. Der folgende Satz formuliert eine hinreichende Bedingung für die Negation dieser Vermutung.

**Satz 15.14** *Falls $\Sigma_k = \Pi_k$ für ein $k \geq 1$, dann kollabiert PH auf den k-ten Level, d.h., dann gilt*

$$PH = \bigcup_{i=0}^{\infty} \Sigma_i = \bigcup_{i=0}^{k} \Sigma_i = \Sigma_k \ .$$

**Beweis** Wir beschränken uns auf die Angabe der zentralen Beweisidee. Die technischen Details sind leicht auszufüllen. Die Klasse $\Sigma_k$ ist im Sinne von Folgerung 15.12 durch eine alternierende Quantorenkette vom Typ $(\exists)_{pol}(\forall)_{pol}(\exists)_{pol}\cdots(Q_k)_{pol}$ gekennzeichnet; die Klasse $\Pi_k$ ist entsprechend durch eine alternierende Quantorenkette vom Typ $(\forall)_{pol}(\exists)_{pol}(\forall)_{pol}\cdots(\bar{Q}_k)_{pol}$ gekennzeichnet. Unter der Voraussetzung $\Sigma_k = \Pi_k$ können wir dann aber auch Sprachen aus $\Pi_k$ mit Hilfe von Quantorenketten vom Typ $(\exists)_{pol}(\forall)_{pol}(\exists)_{pol}\cdots(Q_k)_{pol}$ beschreiben. Demzufolge können wir Sprachen aus $\Sigma_{k+1} = \Sigma'_{k+1} = (\exists)_{pol}[\Pi'_k] = (\exists)_{pol}[\Pi_k]$ auch mit Hilfe von Quantorenketten vom Typ $(\exists)_{pol}(\exists)_{pol}(\forall)_{pol}(\exists)_{pol}\cdots(Q_k)_{pol}$ beschreiben. Durch Verschmelzung der ersten beiden $(\exists)_{pol}$-Quantoren erhalten wir eine Beschreibung vom Typ $(\exists)_{pol}(\forall)_{pol}(\exists)_{pol}\cdots(Q_k)_{pol}$, welche ja (wie bereits erwähnt) gerade Sprachen aus $\Sigma_k$ kennzeichnet. Folglich hat der zusätzliche $(\exists)_{pol}$-Quantor, den Sprachen aus $\Sigma_{k+1}$ zur Verfügung haben, „nichts gebracht“ und es gilt $\Sigma_{k+1} = \Sigma_k$. Aus Dualitätsgründen gilt dann auch $\Pi_{k+1} = \Pi_k$. Die Gleichheit $\Sigma_k = \Pi_k$ auf Stufe $k$ sorgt auf Stufe $k+1$ für $\Sigma_k = \Sigma_{k+1} = \Pi_{k+1}$. Iteration dieses Argumentes (genauer: vollständige Induktion) liefert $\Sigma_k = \Sigma_{k+j} = \Pi_{k+j}$ für alle $j \geq 0$. Es folgt $PH = \cup_{i=0}^{k}\Sigma_k$ und wegen $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \cdots$ natürlich auch $PH = \Sigma_k$. **qed.**

Aus Satz 15.14 ergibt sich ein kleiner Beitrag zu der $P \neq NP$-Debatte:

**Folgerung 15.15** *$P \neq NP \Leftrightarrow P \neq PH$.*

**Beweis** Wegen $P \subseteq NP \subseteq PH$ folgt aus $P \neq NP$ natürlich $P \subset NP \subseteq PH$ und somit auch $P \neq PH$. (Hierbei ist „$\subset$“ das Zeichen für *echte* Inklusion.) Zum Nachweis von $P \neq PH \implies P \neq NP$ genügt es freilich $P = NP \implies P = PH$ zu zeigen. Wie früher bereits nachgewiesen, impliziert $P = NP$ die Aussage $NP =$ co-$NP$. Wegen $NP = \Sigma_1$ und co-$NP = \Pi_1$ ergibt sich mit Satz 15.14 ein Kollaps von $PH$ auf den ersten Level, d.h., $PH = NP$. Zusammen mit der Voraussetzung $P = NP$ ergibt sich nun auch $PH = P$. **qed.**

Folgerung 15.15 bedeutet, dass wir die berühmte $P \neq NP$-Vermutung im Prinzip durch den Nachweis von $P \neq PH$ beweisen könnten. Da $PH$ eine viel mächtigere Komplexitätsklasse ist als $NP$, da sie ja $NP$ bereits auf Stufe 1 zur Teilklasse hat, könnte die Separation von $P$ und $PH$ (durch Nachweis der Existenz einer Sprache in $PH \setminus P$) evtl. leichter zu erreichen sein als die Separation von $NP$ und $P$.

### 15.3.3 Anwendung 3: Vollständige Probleme in PH

Das $NP$-vollständige Problem SAT lässt sich mit einer $(\exists)_{pol}$-quantifizierten Booleschen Formel (in konjunktiver Normalform) beschreiben, da wir nach der Existenz einer erfüllenden

Belegung zu einer gegebenen CNF-Formel fragen. Um zu $\Sigma_k$- oder $\Pi_k$-vollständigen Problemen zu gelangen ist es naheliegend mit quantifizierten Booleschen Formeln zu operieren, wobei die Quantifizierung über eine alternierende Quantorenkette der Länge $k$ erfolgt. Um solche Formeln bequem zu notieren, teilen wir die Booleschen Variablen in $k$ Typen ein:

**Definition 15.16** *Sei $F$ eine Boolesche Formel und $m$ die Anzahl der in $F$ verwendeten Booleschen Variablen. Eine doppelt indizierte Boolesche Variable der Form $v_{ij}$ heiße* Variable vom Typ $i$. *Wir sagen $F$ ist eine Boolesche Formel* mit $k$ Variablentypen, *wenn $m_1, \ldots, m_k$ existieren, so dass $m = m_1 + \cdots + m_k$ und $F$ verwendet genau $m_i$ Variablen vom Typ $i$. Dies seien oBdA die Variablen $\bar{v}_i = (v_{i1}, \ldots, v_{im_i})$ für $i = 1, \ldots, k$. Wir schreiben dann auch $F(\bar{v}_1, \ldots, \bar{v}_k)$ statt $F$, um die Abhängigkeit von den Variablen hervorzuheben.*

Die folgenden Sprachen sind eine naheliegende Verallgemeinerung der Sprache SAT, wobei an die Stelle eines einzelnen $(\exists)_{pol}$-Quantors eine alternierende Quantorenkette tritt und wir nicht mehr darauf bestehen, dass die Formel in konjunktiver Normalform (also eine CNF-Formel) ist.

**Definition 15.17** *Die Sprache $\mathcal{B}_k$ enthalte alle (Kodierungen von) Booleschen Formeln mit $k$ Variablentypen, welche die Bedingung*

$$\exists \bar{a}_1 \in \{0,1\}^{m_1} \forall \bar{a}_2 \in \{0,1\}^{m_2} \exists \bar{a}_3 \in \{0,1\}^{m_3} \ldots Q_k \bar{a}_k \in \{0,1\}^{m_k} : F(\bar{a}_1, \ldots, \bar{a}_k) = 1 \quad (11)$$

*erfüllen.*

**Lemma 15.18** *Für alle $k \geq 1$: $\mathcal{B}_k \in PSpace$.*

**Beweis** $\mathcal{B}_k$ ist ein Teilproblem von TQBF und, wie wir wissen, TQBF $\in PSpace$. **qed.**

Eine zu $\mathcal{B}_k$ duale Klasse ergibt sich wie folgt:

**Definition 15.19** *Die Sprache $\tilde{\mathcal{B}}_k$ enthalte alle (Kodierungen von) Booleschen Formeln mit $k$ Variablentypen, welche die Bedingung*

$$\forall \bar{a}_1 \in \{0,1\}^{m_1} \exists \bar{a}_2 \in \{0,1\}^{m_2} \forall \bar{a}_3 \in \{0,1\}^{m_3} \ldots \bar{Q}_k \bar{a}_k \in \{0,1\}^{m_k} : F(\bar{a}_1, \ldots, \bar{a}_k) = 1 \quad (12)$$

*erfüllen.*

Der folgende Satz ist das Pendant zum Cook'schen Theorem:

**Satz 15.20** *Für alle $k \geq 1$ gilt: $\mathcal{B}_k$ ist $\Sigma_k$-vollständig und $\tilde{\mathcal{B}}_k$ ist $\Pi_k$-vollständig.*

**Beweis** Der Beweis ähnelt dem Beweis des Cook'schen Theorems. Wir skizzieren die wesentlichen Ideen und weisen von Zeit zu Zeit auf die Analogie zum Beweis des klassischen Cook'schen Theorems hin. Genauere technische Details lassen sich leicht einarbeiten.
Wir diskutieren zunächst den Fall einer ungeraden Anzahl $k$ von Variablentypen. In diesem

Fall gilt $Q_k = \exists$, d.h., die alternierende Quantorenkette in (11) endet mit einem $\exists$-Quantor. Sei $L \in \Sigma_k$. Gemäß Folgerung 15.12 gilt (unter Beachtung von $k$ ungerade):

$$L = \{x : (\exists y_1)_{pol}(\forall y_2)_{pol} \cdots (\exists y_k)_{pol} \langle y_1, \ldots, y_k, x\rangle \in L_0\}$$

für eine geeignete Sprache $L_0 \in P$. Wir können oBdA annehmen, dass $y_1, \ldots, y_k$ Wörter über dem Alphabet $\{0,1\}$ sind (in Analogie zu der Annahme binärer Ratestrings bei NTMs). Sei $M_0$ die polynomielle DTM, welche als Akzeptor von $L_0$ auftritt. Wie im Beweis des Cook'schen Theorems lässt sich die polynomielle deterministische Rechnung von $M_0$ auf $\langle y_1, \ldots, y_k, x\rangle$ mit Hilfe einer Booleschen CNF-Formel $F$ beschreiben. $F$ verwendet Boolesche Variable zur Kodierung der Strings $y_1, \ldots, y_k$ und weitere Hilfvariablen. Für $i = 1, \ldots, k$ und $m_i = |y_i|$ seien $\bar{v}_i = (v_{i1}, \ldots, v_{im_i})$ die Booleschen Variablen vom Typ $i$, deren Belegung zum Binärstring $y_i$ korrespondieren soll (in Analogie zu den Booleschen Variablen für die Ratebits im Beweis des Cook'schen Theorems). Die Hilfsvariablen sind alle vom Typ $k$ und dienen (wie im klassischen Beweis des Cook'schen Theorems) der Beschreibung von Rechnungen von $M_0$ auf Eingaben der Form $\langle y_1, \ldots, y_k, x\rangle$. (Es handelt sich also um Variablen, deren Belegungen den jeweiligen Zustand, die jeweilige Kopfposition und die jeweilige Bandinschrift kodieren.) Sei $m'_k$ die Anzahl dieser Hilfsvariablen, welche wir als $\bar{v}'_k = (v'_{k1}, \ldots, v'_{km'_k})$ notieren. Da $M_0$ eine *polynomielle* DTM ist, ist $m'_k$ polynomiell in $m_1, \ldots, m_k, |x|$ beschränkt. Da $m_i = |y_i|$ polynomiell in $|x|$ beschränkt ist, ist dann schließlich auch $m'_k$ wie auch die Gesamtanzahl $m = m_1 + \cdots + m_k + m'_k$ aller in $F$ vorkommender Boolescher Variablen polynomiell in $|x|$ beschränkt. In Analogie zum Beweis des klassischen Cook'schen Theorems lässt sich nun in pol($|x|$) Schritten eine Formel $F(\bar{v}_1, \ldots, \bar{v}_{k-1}, \bar{v}_k, \bar{v}'_k)$ konstruieren, so dass für alle $x \in L$

$$\begin{aligned} x \in L \;&\Leftrightarrow\; (\exists y_1)_{pol}(\forall y_2)_{pol} \cdots (\exists y_k)_{pol} \langle y_1, \ldots, y_k, x\rangle \in L_0 \\ &\Leftrightarrow\; (\exists y_1)_{pol}(\forall y_2)_{pol} \cdots (\exists y_k)_{pol} M_0 \text{ akzeptiert } \langle y_1, \ldots, y_k, x\rangle \\ &\Leftrightarrow\; \exists \bar{a}_1 \in \{0,1\}^{m_1} \forall \bar{a}_2 \in \{0,1\}^{m_2} \ldots \exists \bar{a}_k \in \{0,1\}^{m_k} \exists \bar{a}'_k \in \{0,1\}^{m'_k} : \\ &\qquad F(\bar{a}_1, \ldots, \bar{a}_k, \bar{a}'_k) = 1 \ . \end{aligned}$$

Da wir die letzten beiden $\exists$-Quantoren vor $F(\bar{a}_1, \ldots, \bar{a}_k, \bar{a}'_k)$ verschmelzen können, ist offensichtlich die Abbildung $x \mapsto F$ eine polynomielle Reduktion von $L$ auf $\mathcal{B}_k$. Folglich ist $\mathcal{B}_k$ $\Sigma_k$-vollständig. Die $\Pi_k$-Vollständigkeit von $\tilde{\mathcal{B}}_k$ ergibt sich (für ungerades $k$) durch Dualisierung.
Bleibt der Fall eines geraden $k$ zu diskutieren. Hier ist es technisch leichter zunächst die $\Pi_k$-Vollständigkeit von $\tilde{B}_k$ zu verifizieren und die $\Sigma_k$-Vollständigkeit von $B_k$ durch Dualisierung zu folgern. Es ist nämlich praktischer, wenn die alternierende Quantorenkette erneut mit einem $\exists$-Quantor endet, damit der $\exists$-Quantor für die Hilfsvaribalen der Booleschen Formel $F$ mit dem vorangehenden $\exists$-Quantor verschmolzen werden kann. Der Nachweis der$\Pi_k$-Vollständigkeit von $\tilde{B}_k$ für gerades $k$ geschieht analog zum Nachweis der $\Sigma_k$-Vollständigkeit von $\mathcal{B}_k$ für ungerades $k$. **qed.**

Eine genauere Inspektion des Beweises von Satz 15.20 führt zu **folgender Beobachtung**: **Übg.**

**Folgerung 15.21** *Die Einschränkung von $\mathcal{B}_k$ auf CNF-Formeln bei ungeradem $k$ und DNF-Formeln bei geradem $k$ ist $\Sigma_k$-vollständig. Dual dazu ist die Einschränkung von $\tilde{B}_k$ auf DNF-Formeln bei ungeradem $k$ und CNF-Formeln bei geradem $k$ $\Pi_k$-vollständig.*

Folgerung 15.21 liefert im Falle $k = 1$ gerade das Cook'sche Theorem.

Weiterhin ergibt sich aus $\mathcal{B}_k \in \mathit{PSpace}$ und der $\Sigma_k$-Härte von $\mathcal{B}_k$ unmittelbar die

**Folgerung 15.22** *$PH \subseteq PSpace$.*

### Offene Probleme im Zusammenhang mit PH

**P versus PH** Sind prädikatenlogische Aussagen, die eine alternierende Quantorenkette[9] konstanter Länge verwenden dürfen, mächtiger als rein aussagenlogische Aussagen?

**Kollaps versus Echtheit von PH** Sind alternierende Quantorenketten der Länge $k + 1$ stets mächtiger als alternierende Quantorenketten der Länge $k$?

**PH versus PSpace** Sind alternierende Quantorenketten variabler Länge mächtiger als alternierende Quantorenketten konstanter Länge?

[9] Gemeint sind stets Ketten, die zwischen dem $(\exists)_{pol}$- und dem $(\forall)_{pol}$-Quantor abwechseln.