# 11

# Linearzeit-Temporale Logik

In diesem Kapitel betrachten wir noch eine weitere Logik, die sich einerseits von der Art ihrer Operatoren her wesentlich von MSO unterscheidet, sich andererseits jedoch in MSO—sogar in FO—einbetten lässt. Der Unterschied zu MSO besteht hauptsächlich darin, dass es keine direkten Quantoren über Positionen und Mengen von Positionen in einem Wort gibt. Stattdessen stellt man sich im Rahmen der Temporallogik ein Wort als einen zeitlichen Ablauf von Ereignissen, die durch die Alphabetsymbole kodiert sind, vor. Eine temporallogische Formel wird dann immer in einer einzigen Position des Wortes interpretiert. Diese stellt sozusagen den jetzigen Zeitpunkt dar, während das Suffix des Wortes an dieser Position die Zukunft repräsentiert. Die Operatoren der temporalen Logik machen dann Aussagen über Ereignisse in der Zukunft. Hier beschränken wir uns auf eine der einfachsten Temporallogiken, das sogenannte LTL.

Die Sichtweise des zeitlichen Ablaufens eines Wortes legt eine Beziehung zu Automaten nahe. Diese Nähe zeichnet sich hier dadurch aus, dass die Übersetzung von LTL in Büchi-Automaten einfacher ist und vor allem effizienter ist, als das über die Codierung in MSO der Fall wäre.

LTL wird auch häufig als Spezifikationssprache für reaktive Systeme eingesetzt. Eines der fundamentalen Probleme, welches in diesem Gebiet auftritt, ist, zu einer Beschreibung eines reaktiven Systems $\mathcal{T}$ und einer gegebenen Spezifikation $\varphi$ zu entscheiden, ob $\mathcal{T}$ die Anforderung $\varphi$ erfüllt. Auch dort erweisen sich Büchi-Automaten wiederum als geeignetes Hilfsmittel für den algorithmischen Zugang zu einem logischen Problem.

## 11.1 Syntax und Semantik

Wir erinnern noch einmal an die Übersetzung einer WMSO-Formel in einen NFA aus dem Kapitel 2. Eine WMSO-Formel $\varphi$ mit freien Variablen $X_1, \ldots, X_n$ —zur Vereinfachung nehmen wir an, dass diese nur freie zweitstufige Variablen enthält—wurde dort in einen NFA übersetzt, der über dem Alphabet $\{0,1\}^n$

arbeitet. Ein Wort ist also zusammengesetzt aus Vektoren der Länge $n$. Dies kann man aber auch als Wort mit $n$ Spuren ansehen. Die $i$-te Spur kodiert dann die Belegung der Variablen $X_i$—eine 1 steht genau an den Positionen, die zu der Menge gehören, die $X_i$ interpretiert.

Bei den weiteren Untersuchungen an Automaten in diesem Buch sind wir meist davon ausgegangen, dass das Alphabet abstrakt als Menge gegeben ist. Dies schließt natürlich den Fall der $n$-stelligen Vektoren mit ein. Hier kommen wir jetzt wieder darauf zurück. In einer LTL-Formel gibt es ebenfalls zweitstufige Variablen. Traditionell bezeichnet man diese jedoch als *Propositionen* und schreibt Kleinbuchstaben $p, q, \ldots$ anstatt der bisher üblichen Großbuchstaben $X, Y, \ldots$. Wir werden dies hier praktisch genauso handhaben. Es gilt wiederum, dass eine endliche Menge $\mathcal{P}$ von Propositionen oder zweitstufigen Variablen ein Alphabet $\Sigma = 2^{\mathcal{P}}$ definiert. Es bietet sich jedoch der Einfachheit halber an, solch ein Alphabetsymbol als Teilmenge der Propositionen anstatt als Vektor aufzufassen. So kann man z.B. schlicht $p \in a_i$ schreiben anstatt eine neue Notation für den Zugriff auf die $p$-te Komponente des Vektors $a_i$ einführen zu müssen. Mathematisch sind diese beiden Sichtweise jedoch natürlich äquivalent. Die Teilmenge besteht genau als denjenigen Propositionen, an deren Stelle im Vektor eine 1 stehen würde.

**Definition 11.1.** Sei $\mathcal{P} = \{p, q, \ldots\}$ eine Menge von Propositionen. Formeln der *Linearzeit-Temporallogik* (LTL) über $\Sigma$ sind gegeben durch die folgende Grammatik.

$$\varphi \quad ::= \quad p \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi \mathtt{U} \varphi$$

wobei $p \in \mathcal{P}$. Wir benutzen neben den üblichen Abkürzungen der Aussagenlogik auch die folgenden: $\varphi \mathtt{R} \psi := \neg(\neg\varphi \mathtt{U} \neg\psi)$, $\mathtt{F}\varphi := \mathit{true}\mathtt{U}\varphi$ und $\mathtt{G}\varphi := \neg\mathtt{F}\neg\varphi$.

Sei $w = a_0 a_1 \ldots \in (2^{\mathcal{P}})^\omega$. Die Semantik einer LTL-Formel ist induktiv definiert für alle $i \in \mathbb{N}$ wie folgt.

$$\begin{array}{lcl}
w, i \models p & \iff & p \in a_i \\
w, i \models \varphi \vee \psi & \iff & w, i \models \varphi \text{ oder } w, i \models \psi \\
w, i \models \neg\varphi & \iff & w, i \not\models \varphi \\
w, i \models \bigcirc\varphi & \iff & w, i+1 \models \varphi \\
w, i \models \varphi \mathtt{U} \psi & \iff & \text{es gibt } k \geq i, \text{ so dass } w, k \models \psi \\
 & & \text{und für alle } j : i \leq j < k \Rightarrow w, j \models \varphi
\end{array}$$

Wie oben bereits erwähnt, wird eine LTL-Formel also an einer Position in einem Wort interpretiert. Sinnvolle Aussagen über solche Positionen sind z.B. "die aktuelle Position gehört zu $p$, aber nicht zu $q$", was durch die Formel $p \wedge \neg q$ ausgedrückt wird.

Der Operator $\bigcirc$ ließt sich als "*in der nächsten Position*". Die Formel $p \wedge \bigcirc q$ drückt dann aus, dass an der aktuellen Position (u.a.) ein $p$ steht, welches von einem $q$ gefolgt wird.

Der Operator $\mathtt{U}$ steht für "*until*". Die Formel $\varphi \mathtt{U} \psi$ gilt in einer aktuellen Position eines Wortes, wenn es in der Zukunft (inklusive des momentanen

Zeitpunkts) eine Position gibt, in der $\psi$ gilt, so dass alle Positionen davor (wiederum inklusive des momentanen Zeitpunkts) $\varphi$ erfüllen. Damit also $\varphi \mathtt{U} \psi$ zu einem Zeitpunkt $t$ gilt, muss entweder $\psi$ zum Zeitpunkt $t$ gelten, oder aber $\varphi$ muss zum Zeitpunkt $t$ und allen folgenden Zeitpunkten bis (ausschließlich) zum erstmaligen Auftreten von $\psi$. Insbesondere muss $\psi$ auch tatsächlich irgendwann auftreten.

Ein Spezialfall davon ist das sogenannte "*finally*" `F`. Hier ist das linke Argument des `U`-Operators *true*, d.h. an die dazwischenliegenden Zeitpunkte werden keine Anforderungen gestellt. Daher bedeutet $\mathtt{F}\varphi$, dass irgendwann in der Zukunft oder bereits jetzt $\varphi$ gilt.

Der zu `F` duale Operator ist das "*generally*" `G`. Die Formel $\mathtt{G}\varphi$ bedeutet, dass ab sofort $\varphi$ immer gilt. Man überlegt sich leicht, dass dies gleichbedeutend damit ist, dass es keinen Moment in der Zukunft gibt, in dem $\varphi$ nicht gilt—daher die Dualität zum `F`-Operator.

Das Duale zum allgemeineren `U` ist das "*release*" `R`. Die Formel $\varphi \mathtt{R} \psi$ besagt, dass $\psi$ gelten muss, solange es nicht von $\varphi$ abgelöst wird. Somit sind Modelle dieser Formel solche, in denen ab dem aktuellen Moment entweder $\psi$ immer gilt, oder $\psi$ gilt bis zu einem Moment, in dem sowohl $\psi$ als auch $\varphi$ gelten.

Auf natürliche Art und Weise, genauer gesagt durch Interpretation einer Formel in der ersten Position eines Wortes, lässt sich einer LTL-Formel eine Sprache unendlicher Wörter zuordnen.

**Definition 11.2.** Sei $\Sigma = 2^{\mathcal{P}}$, $w \in \Sigma^\omega$ und $\varphi \in \mathrm{LTL}$. Wir schreiben $w \models \varphi$ falls $w, 0 \models \varphi$, und $L(\varphi) := \{w \mid w \models \varphi\}$.

*Beispiel 11.3.* In LTL lässt sich "unendlich oft gilt $\varphi$" durch $\mathtt{GF}\varphi$ ausdrücken. Dies besagt, dass immer irgendwann—also immer wieder—$\varphi$ gilt. Genauso bedeutet $\mathtt{FG}\neg\varphi$ dann, dass $\varphi$ nur endlich oft gilt.

Die Sprache $\{w \in (2^{\{p,q\}})^\omega \mid$ wenn unendliche viele Positionen zu $p$ gehören, dann auch unendlich viele zu $q\}$ ist LTL-definierbar. Sie wird durch die Formel $\mathtt{GF}q \vee \mathtt{FG}\neg p$ beschrieben.

Setzt man die Alphabetsymbole aus $\Sigma$ als gegeben voraus, so stellt sich die Frage, wie man z.B. eine umgangssprachliche Beschreibung einer formalen Sprache über $\Sigma$ in eine LTL-Formel umsetzt. Dazu kann man *charakteristische Formeln* $\chi_a$ verwenden. Sei wiederum $\Sigma = 2^{\mathcal{P}}$ und $\mathcal{P}$ endlich. Dann lässt sich ein Alphabetsymbol $a \in \Sigma$, also $a \subseteq \mathcal{P}$, durch Aufzählung derjenigen Propositionen, die in ihm enthalten sind, sowie derjenigen, die dies nicht sind, eindeutig beschreiben.

$$\chi_a \;:=\; \Big(\bigwedge_{p \in a} p\Big) \wedge \Big(\bigwedge_{p \notin a} \neg p\Big)$$

*Beispiel 11.4.* Die Sprache, die nur aus dem Wort $(ab)^\omega$ besteht, wird z.B. durch die LTL-Formel

$$\chi_a \wedge \mathtt{G}\big((\chi_a \to \bigcirc \chi_b) \wedge (\chi_b \to \bigcirc \chi_a)\big)$$

beschrieben.

Wir bemerken ohne Beweis, dass die Sprache $(a(a \cup b))^\omega$, also die Menge aller Wörter, in denen an jeder geraden Stelle ein $a$ steht, nicht LTL-definierbar ist. Dies hängt damit zusammen, dass LTL-definierbare Sprachen bereits auch in FO auf unendlichen Wörtern definierbar sind. Zur Erinnerung: Auf endlichen Wörtern lässt sich in FO nicht die Sprache aller Wörter gerader Länge definieren. Das Prinzip hinter der Nichtausdrückbarkeit obiger Sprache ist genau dasselbe.

Die Semantik einer LTL-Formel ist bereits in FO aufgeschrieben. Genauer gesagt lässt sich durch Formalisierung der semantischen Gleichungen zu jeder LTL-Formel $\varphi$ eine FO-Formel $\hat{\varphi}(x)$ angeben, so dass für alle $i \in \mathbb{N}$ gilt: $w, i \models \varphi \iff w, i \models \hat{\varphi}(x)$. Somit gilt $w \models \varphi \iff w \models \hat{\varphi}(min)$. Daraus ergibt sich insbesondere, dass jede in LTL definierbare Sprache auch in FO definierbar ist.

**Definition 11.5.** Eine LTL-Formel ist in *positiver Normalform*, wenn sie nur aus Literalen $p, \neg p$ mit $p \in \mathcal{P}$ und den Operatoren $\vee, \wedge, \bigcirc, \mathtt{U}$ und $\mathtt{R}$ aufgebaut ist.

**Lemma 11.6.** *Jede LTL-Formel $\varphi$ über einem Alphabet $\Sigma$ ist äquivalent zu einer LTL-Formel $\varphi'$ in positiver Normalform, so dass $|\varphi'| \leq 2 \cdot |\varphi|$ gilt.*

*Beweis.* Übung.

In den folgenden Übersetzungen von LTL-Formeln in Automaten auf unendlichen Wörtern spielen die sogenannten *Abwicklungen* der temporalen Operatoren $\mathtt{U}$ und $\mathtt{R}$ eine große Rolle. Was damit gemeint ist, wird anhand des $\mathtt{U}$-Operators im folgenden Lemma erklärt.

**Lemma 11.7.** *Für alle $\varphi, \psi \in LTL$ gilt: $\varphi \mathtt{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi))$.*

*Beweis.* Wir zeigen hier nur die Richtung "$\Rightarrow$". Die Umkehrung ist vollkommen analog.

Sei $w, i \models \varphi \mathtt{U} \psi$. Dann existiert ein $k \geq i$ mit $w, i \models \psi$ und $w, j \models \varphi$ für alle $j$ mit $i \leq j < k$. Wir unterscheiden zwei Fälle.

a) Fall 1, $k = i$. Dann gilt also offensichtlich $w, i \models \psi$.
b) Fall 2, $k > i$. Dann muss erstens $w, i \models \varphi$ gelten. Außerdem betrachten wir die Position $i + 1$ in $w$. Da nach Annahme $k \geq i + 1$ ist, $w, k \models \psi$ gilt und auch noch $w, h \models \varphi$ für alle $h$ mit $i \leq h < k$, also insbesondere auch für alle $h$ mit $i + 1 \leq h < k$ gilt, gilt ebenfalls $w, i + 1 \models \varphi \mathtt{U} \psi$. Dann gilt aber auch $w, i \models \bigcirc(\varphi \mathtt{U} \psi)$. Zusammen mit der ersten Beobachtung haben wir also $w, i \models \varphi \wedge \bigcirc(\varphi \mathtt{U} \psi)$.

Da nun immer einer der beiden Fälle eintreten muss, gilt auch $w, i \models \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi))$. □

Äquivalenzen in LTL sind sogar Kongruenzen, d.h. falls $\varphi \equiv \psi$, so gilt auch $\theta(\varphi) \equiv \theta(\psi)$ für jeden Kontext $\theta$, in den $\varphi$ und $\psi$ eingebettet werden. Daraus folgt sofort folgendes.

$$\begin{aligned}
\varphi \mathtt{U} \psi &\equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi)) \\
&\equiv \psi \vee (\varphi \wedge \bigcirc(\psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi)))) \\
&\equiv \psi \vee (\varphi \wedge \bigcirc(\psi \vee (\varphi \wedge \bigcirc(\psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi)))))) \\
&\equiv \ldots
\end{aligned}$$

Dies liefert auch eine Methode, um von einem Wort zu zeigen, dass es an einer Stelle $i$ die Formel $\varphi \mathtt{U} \psi$ erfüllt. Entweder man zeigt, dass an Stelle $i$ bereits $\psi$ gilt. Oder man zeigt, dass an ihr $\varphi$ gilt und dass an Stelle $i+1$ wiederum $\varphi \mathtt{U} \psi$ gilt. Fasst man dies als Spiel auf, so kann es natürlich unendliche Partien geben. Dies kann aber nur dadurch geschehen, dass man an keiner Stelle $k$ mit $k \geq i$ irgendwann einmal $\psi$ bewiesen hat, sondern an jeder solchen Stelle $\varphi$ bewiesen hat und zur nächsten Stelle übergegangen ist. Somit hat man aber nicht gezeigt, dass die Formel $\varphi \mathtt{U} \psi$ an der Stelle $i$ gilt, denn dies verlangt die Existenz einer Position $k \geq i$, an der $\psi$ gilt.

Jedes Entscheidungsverfahren für LTL, welches die temporalen Operatoren—insbesondere das `U` auf diese Art und Weise verarbeitet, muss sicherstellen, dass jede Formel der Form $\varphi \mathtt{U} \psi$ nur endlich oft abgewickelt wird, dass man also irgendwann wirklich einmal $\psi$ auswählt, anstatt die Entscheidung immer nur weiter zu verschieben.

Die in den beiden folgenden Abschnitten vorgestellten Verfahren zur Konstruktion von Büchi-Automaten aus LTL-Formeln benutzen implizit diese Abwicklungen für `U`-Formeln (und die analoge Variante für `R`-Formeln). Über eine geschickte Wahl der Endzustände wird sichergestellt, dass unendliche Abwicklungen einer `U`-Formel nicht dazu führen, dass ein Wort akzeptiert wird.

## 11.2 LTL und nichtdeterministische Büchi-Automaten

Um eine direkte Übersetzung von LTL in Büchi-Automaten ohne den Umweg über FO und MSO anzugeben, bietet es sich an, zunächst als technisches Hilfsmittel die Definition eines Büchi-Automaten zu verallgemeinern.

### 11.2.1 Verallgemeinerte Büchi-Automaten

**Definition 11.8.** Ein *verallgemeinerter Büchi-Automat* (GNBA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, I, \delta, F_1, \ldots, F_k)$ wie bei einem NBA, jedoch mit Anfangszustandsmenge $I \subseteq Q$ und mehreren Endzustandsmengen $F_1, \ldots, F_k \subseteq Q$. Ein Lauf eines solchen Automaten auf einem Wort $w \in \Sigma^\omega$ ist definiert wie bei einem NBA, mit dem Unterschied, dass er in einem beliebigen Zustand $q \in I$ anfängt.

Solch ein Lauf $q_0, q_1, \ldots$ heißt *akzeptierend*, falls für alle $i = 1, \ldots, k$ ein $q \in F_i$ gibt, so dass $q = q_j$ für unendlich viele $j$.

Ein verallgemeinerter Büchi-Automat hat also erstens eine Anfangszustandsmenge statt einem einzigen Anfangszustand und zweitens mehrere Endzustandsmengen, die jeweils unendlich oft besucht werden müssen.

**Satz 11.9.** *Für jeden GNBA* $\mathcal{A} = (Q, \Sigma, I, \delta, F_0, \ldots, F_{k-1})$ *gibt es einen Büchi-Automaten* $\mathcal{A}'$*, so dass* $L(\mathcal{A}') = L(\mathcal{A})$ *und* $|\mathcal{A}'| = 1 + |Q| \cdot (k+1)$.

*Beweis.* Wir beschränken uns hier darauf zu zeigen, wie man zu solch einem GNBA einen GNBA $\mathcal{A}'$ macht, der nur noch eine einzige Endzustandsmenge besitzt. Die Reduzierung mehrere Anfangszustände zu einem ist dann sehr einfach durch Hinzunahme eines neuen Zustands zu machen.

Definiere $\mathcal{A}'$ als $(Q \times \{0, \ldots, k-1\}, \Sigma, I \times \{0\}, \Delta, F_0 \times \{0\}$. Die Transitionsfunktion ist wie folgt definiert.

$$\Delta\big((q,i),a\big) \;:=\; \{(p,j) \mid p \in \delta(q,a) \text{ und } j = \begin{cases} i+1 \mod k & \text{, falls } q \in F_i \\ i & \text{, sonst} \end{cases}$$

Der Zustandsmenge des GNBA wird also ein Zähler angehängt, der angibt, aus welcher ursprünglichen Endzustandsmenge als nächstes ein Endzustand durchlaufen werden soll. Dies beruht auf dem Prinzip, dass in einem Lauf von $\mathcal{A}$ aus allen $F_i$ unendlich oft Zustände gesehen werden genau dann, wenn irgendwann einer aus $F_0$ gesehen wird und jedes Mal, wenn einer aus $F_i$ durchlaufen wird, danach auch irgendwann einer aus $F_{i+1 \mod k}$ gesehen wird. Aus dieser Überlegung und der Tatsache, dass $\mathcal{A}'$ in seiner ersten Zustandskomponente lediglich $\mathcal{A}$ simuliert, ergibt sich sofort die Korrektheit dieser Konstruktion. □

Die Wahl der 0 in der zweiten Komponenten der Anfangs- und Endzustände von $\mathcal{A}'$ ist willkürlich. Genauso könnte man auch jede andere Zahl aus $\{0, \ldots, k-1\}$ wählen. Wichtig ist nur, dass die Endzustände von der Form $F_i \times \{i\}$ sind. Dieser Index muss jedoch noch nicht einmal mit dem in den Anfangszuständen übereinstimmen.

### 11.2.2 Von LTL nach NBA

Im Folgenden ist immer davon auszugehen, dass LTL-Formeln in positiver Normalform vorliegen. Wir betrachten zunächst eine rein syntaktische Definition, die das Konzept der Unterformelmenge leicht erweitert.

**Definition 11.10.** Der *Fischer-Ladner-Abschluss* einer LTL-Formel $\theta$ in positiver Normalform ist die kleinste Menge $FL(\theta)$, die $\theta$ enthält und für die folgendes gilt.

- $\varphi \vee \psi \in FL(\theta) \Rightarrow \{\varphi, \psi\} \subseteq FL(\theta)$,
- $\varphi \wedge \psi \in FL(\theta) \Rightarrow \{\varphi, \psi\} \subseteq FL(\theta)$,
- $\bigcirc\varphi \in FL(\theta) \Rightarrow \varphi \in FL(\theta)$,

- $\varphi \mathtt{U} \psi \in FL(\theta) \Rightarrow \{\varphi, \psi, \bigcirc(\varphi \mathtt{U} \psi), \varphi \wedge \bigcirc(\varphi \mathtt{U} \psi), \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi))\} \subseteq FL(\theta)$,
- $\varphi \mathtt{R} \psi \in FL(\theta) \Rightarrow \{\varphi, \psi, \bigcirc(\varphi \mathtt{R} \psi), \varphi \vee \bigcirc(\varphi \mathtt{R} \psi), \psi \wedge (\varphi \vee \bigcirc(\varphi \mathtt{R} \psi))\} \subseteq FL(\theta)$,

Darauf aufbauend betrachten wir nun eine semantische Definition, nämlich die einer Hintikka-Menge. Solche Mengen kann man sich intuitiv vorstellen als abgeschlossen unter "dem was auch noch gelten muss". Enthält solch eine Menge z.B. eine Disjunktion, die irgendwo gelten soll, dann muss sicherlich eines der Disjunkte ebenfalls gelten. Bei Konjunktionen sind es sogar beide Konjunkte.

**Definition 11.11.** Eine *Hintikka-Menge* für eine LTL-Formel $\theta$ in positiver Normalform ist eine Menge $M \subseteq FL(\theta)$, für die gilt:

- $\varphi \vee \psi \in M \Rightarrow \varphi \in M$ oder $\psi \in M$,
- $\varphi \wedge \psi \in M \Rightarrow \varphi \in M$ und $\psi \in M$,
- $\varphi \mathtt{U} \psi \in M \Rightarrow \psi \in M$ oder $\big(\ \varphi \in M$ und $\bigcirc(\varphi \mathtt{U} \psi) \in M\ \big)$,
- $\varphi \mathtt{R} \psi \in M \Rightarrow \psi \in M$ und $\big(\ \varphi \in M$ oder $\bigcirc(\varphi \mathtt{R} \psi) \in M\ \big)$.

Eine Hintikka-Menge $M$ heißt *konsistent*, falls es keine $p \in \mathcal{P}$ gibt mit $\{p, \neg p\} \subseteq M$. Mit $\mathcal{H}(\theta)$ bezeichnen wir die Menge aller konsistenten Hintikka-Mengen über $\theta$.

Mit $\mathcal{P}^+(M)$ bezeichnen wir die Menge aller positiven vorhandenen Propositionen in $M$, also lediglich $M \cap \mathcal{P}$; mit $\mathcal{P}^-(M)$ bezeichnen wir die Menge der negativ vorhandenen Propositionen darin, also $\{p \mid p \in \mathcal{P} \text{ und } \neg p \in M\}$.

Sei $\theta$ nun einmal fest gewählt. Wir konstruieren daraus einen GNBA $\mathcal{A}_\theta$, der genau die Menge aller Modelle von $\theta$ erkennt. Als Zustände benutzen wir konsistente Hintikka-Mengen über $\theta$. Solche Mengen beschreiben intuitiv, welche Unterformeln—genauer: Elemente des Fischer-Ladner-Abschlusses—an einer Stelle in einem Modell gelten müssen. Wir benutzen hier den Nichtdeterminismus in den Büchi-Automaten, um diese in jedem Schritt erraten zu lassen. Die Konsistenz der Mengen sorgt dafür, dass die Automaten in einem Schritt nichts raten, was aus aussagenlogischen Gründen bereits ausgeschlossen ist. Für die temporalen Operatoren, insbesondere das `U` und `R` benutzen wir die Charakterisierungen über deren Abwicklungen aus dem vorigen Abschnitt. Diese sind ja syntaktisch bereits in die Definitionen von Fischer-Ladner-Abschluss und Hintikka-Menge eingeflossen. Die `U`-Formeln spielen dabei eine spezielle Rolle, denn man darf das Erfüllen einer $\varphi \mathtt{U} \psi$-Formel nicht beliebig lange hinauszögern. Dies werden wir durch die Akzeptanzbedingung des GNBA sicherstellen.

Seien $\varphi_1 \mathtt{U} \psi_1, \ldots, \varphi_k \mathtt{U} \psi_k$ alle in $FL(\theta)$ vorkommenden `U`-Formeln. Wir konstruieren den GNBA $\mathcal{A}_\theta := (\mathcal{H}(\varphi), \Sigma, I, \delta, F_1, \ldots, F_k)$, wobei

- die Anfangszustandsmenge aus allen konsistenten Hintikka-Mengen besteht, die die ursprüngliche Formel $\theta$ enthalten:

$$I \;=\; \{M \mid \theta \in M\}$$

- die $i$-te Endzustandsmenge, $i = 1, \ldots, k$, diejenigen Zustände enthält, in denen die $i$-te $\mathtt{U}$-Formel erfüllt wird, falls sie noch erfüllt werden muss:

$$F_i \;=\; \{M \mid \varphi_i \mathtt{U} \psi_i \in M \Rightarrow \psi_i \in M\}$$

- die Übergangsrelation wie folgt definiert ist.

$$\delta(M,a) = \begin{cases} \{M' \mid \text{ für alle } \bigcirc\psi \in M : \psi \in M'\}, & \text{falls } \mathcal{P}^+(M) \subseteq a \text{ und} \\ & \quad \mathcal{P}^-(M) \not\subseteq a \\ \emptyset, & \text{sonst} \end{cases}$$

Die Funktionsweise des GNBA $\mathcal{A}_\theta$ ist also intuitiv die folgende. Da er genau die Modelle der Formel $\theta$ erkennen soll, errät er zunächst eine Hintikka-Menge, die $\theta$ enthält. Aus der Hintikka-Mengen-Eigenschaft folgt, dass damit in dieser Menge auch eine Auswahl an allen Formeln enthalten ist, die wegen $\theta$ ebenfalls in Position 0 eines vorgelegten Wortes gelten müssen.

Ist $\mathcal{A}_\theta$ einmal in einem Zustand $M$ angelangt, so enthält dieser insbesondere Literale $p$ oder $\neg q$ und Formeln der Form $\bigcirc\psi$, für die es keine Regeln in der Definition einer Hintikka-Menge mehr gibt. Diese Regeln sorgen nur dafür, dass man auch weitere Formeln aufnimmt, die in *derselben* Position gelten müssen. Eine Formel der Form $\bigcirc\psi$ macht aber eine Aussage über die *nachfolgende* Position. Macht der Automat nun einen Transitionsschritt, so kann dies nur mit solch einem Alphabetsymbol geschehen, welches mit den aktuell vorhandenen Literalen konsistent ist. Dies bedeutet, dass das Alphabetsymbol alle positiv vorhandenen Literale, jedoch keines der negativ vorhandenen Literale enthalten sollte.

Nach solch einem Schritt muss er natürlich eine Hintikka-Menge erreichen, die alle diejenigen Formeln enthält, von denen er zuvor geraten hat, dass sie im nächsten Schritt gelten müssen. Der Automat muss also auch in Bezug auf die $\bigcirc$-Formeln konsistent bleiben.

Wir betrachten zunächst den Automaten für die einfache Formel $p\mathtt{U}\neg p$. Dieser hat zwei Anfangszustände, nämlich $M_0 = \{p\mathtt{U}\neg p, \neg p\}$ und $M_1 = \{p\mathtt{U}\neg p, p, \bigcirc(p\mathtt{U}\neg p)\}$. Man sieht leicht, dass $M_1 \in \delta(M_1, a)$ gilt, solange $p \in a$ ist. Diese Transition entspricht einem einmaligen Abwickeln der $\mathtt{U}$-Formel, ohne diese danach zu erfüllen: Er erwartet noch ein Symbol $a$, welches $p$ enthält, und will im nächsten Schritt weiterhin $p\mathtt{U}\neg p$ beweisen. Dies ist lokal richtig, im Globalen muss jedoch dafür gesorgt werden, dass diese Transition nicht immer nur genommen wird. Ansonsten würde ja das Wort $\{p\}^\omega$ akzeptiert, welches sicherlich nicht Modell von $p\mathtt{U}\neg p$ ist. Die Akzeptanzzustände sind $M_0$ und $\emptyset$. Der GNBA ist hier bereits ein NBA, da die Ursprungsformel nur ein einziges $\mathtt{U}$ enthält.

**Satz 11.12.** *Sei $\theta$ LTL-Formel in positiver Normalform. Dann gilt $L(\mathcal{A}_\theta) = L(\theta)$ und $|\mathcal{A}_\theta| \leq 2^{2|\theta|}$.*

*Beweis.* Beachte: $FL(\theta) \leq 2|\theta|$, da es zu jeder Unterformel von $\theta$ höchstens 2 Formeln im Fischer-Ladner-Abschluss geben kann. Genauer gesagt enthält

der Fischer-Ladner-Abschluss jede Unterformel, und für jede U-Formel darin noch höchstens eine weitere, in der dieser ein $\bigcirc$ vorangestellt wird. Somit gilt $|\mathcal{H}(\theta)| \leq 2^{2|\theta|}$, was die Behauptung über die Größe des resultierenden GNBA zeigt.

Es bleibt noch die Korrektheit zu zeigen. Wir nehmen wieder an, dass $\varphi_1 \mathtt{U} \psi_1, \ldots, \varphi_k \mathtt{U} \psi_k$ alle U-Formeln in $FL(\theta)$ sind.

"$\supseteq$" Sei $w \in L(\theta)$. Definiere für jedes $i \in \mathbb{N}$ eine Menge $M_i := \{\psi \in FL(\theta) \mid w, i \models \psi\}$. Beachte Folgendes.

a) Für alle $i \in \mathbb{N}$ gilt $M_i \in \mathcal{H}(\theta)$, denn die Menge aller Formeln, die in einer Position eines Wortes gelten, bildet immer eine Hintikka-Menge.
b) Nach Voraussetzung gilt $\theta \in M_0$.
c) Für alle $i \in \mathbb{N}$ gilt: Wenn $\bigcirc\psi \in M_i$, dann $\psi \in M_{i+1}$.
d) Falls der $i$-te Buchstabe von $w$ ein $a$ ist, dann ist $M_i \cap \Sigma \subseteq \{a\}$.
e) Für jedes $j \in \{1, \ldots, k\}$ und jedes $i \in \mathbb{N}$ gilt: wenn $\varphi_j \mathtt{U} \psi_j \in M_i$ dann existiert ein $i' \geq i$, so dass $\psi_j \in M_{i'}$. Wenn also an einer Stelle eine U-Formel gilt, dann muss diese auch irgendwann erfüllt sein dadurch, dass ihr rechtes Argument irgendwann ebenfalls gilt.

Wegen (1) bildet $M_0, M_1, \ldots$ eine Sequenz von Zuständen von $\mathcal{A}_\varphi$. Wegen (2) beginnt diese in einem Anfangszustand. Wegen (3) und (4) ist diese Sequenz ein gültiger Lauf, der die Transitionsrelation befolgt und der niemals stehen bleibt. Und wegen (5) ist er akzeptierend.

"$\subseteq$" Sei $w \in L(\mathcal{A}_\varphi)$. D.h. es existiert ein akzeptierender Lauf $M_0, M_1, \ldots$ von $\mathcal{A}$ auf $w$. Man zeigt nun leicht durch Induktion über den Formelaufbau, dass für alle $i \in \mathbb{N}$ und alle $\psi \in FL(\varphi)$ folgendes gilt. Wenn $\psi \in M_i$ dann $w, i \models \psi$.

Im Induktionsanfang sei $\psi = p$ für ein $p \in \mathcal{P}$. Nach Voraussetzung gilt $p \in M_i$. Man beachte, dass $\mathcal{A}_\theta$ die Transition von $M_i$ zu $M_{i+1}$ nur machen kann, wenn an der $i$-ten Stelle im Wort dann auch ein Buchstabe $a$ steht, so dass $p \in a$ gilt. Damit gilt dann aber auch $w, i \models p$.

Im Induktionsschritt für die Boole'schen Operatoren folgt die Aussage aus der Hypothese und der Tatsache, dass jedes $M_i$ eine Hintikka-Menge bildet. Sei also $\psi = \bigcirc\psi'$. Da $\psi \in M_i$, haben wir $\psi' \in M_{i+1}$. Nach der Induktionshypothese gilt $w, i+1 \models \psi'$ und somit $w, i \models \psi$.

Es verbleiben die zwei interessantesten Fälle der temporalen Operatoren U und R.

Sei nun $\varphi_j \mathtt{U} \psi_j \in M_i$ für ein $i \in \mathbb{N}$. Da $M_i$ eine konsistente Hintikka-Menge ist, gilt entweder $\psi_j \in M_i$, woraus die Behauptung sofort per Induktionshypothese folgt. Oder aber es gilt $\varphi_j \in M_j$ und $\bigcirc(\varphi_j \mathtt{U} \psi_j) \in M_i$. Nach der Definition der Transitionsrelation ist dann aber $\varphi_j \mathtt{U} \psi_j \in M_{i+1}$. Dieses Argument lässt sich nun iterieren, was $w, i' \models \varphi_j$ für $i' = i, i+1, i+2, \ldots$ zeigt. Man beachte außerdem, dass $\varphi_j \mathtt{U} \psi_j \in M_{i'}$ für $i' = i, i+1, i+2, \ldots$ gilt. Da der zugrundeliegende Lauf aber akzeptierend ist, muss es ein $i' > i$ geben, so dass $\psi_j \in M_{i'}$. Anderenfalls würde dieser Lauf keinen Endzustand aus der

zu $\varphi_j \mathtt{U} \psi_j$ gehörenden Endzustandsmenge mehr sehen und somit nicht die verallgemeinerte Büchi-Bedingung erfüllen. Die Induktionshypothese, angewandt auf $\psi_j$ liefert dann $w, i' \models \psi_j$. Angewandt auf $\varphi_j$ liefert sie $w, h \models \varphi_j$ für alle $i \leq h < i'$. Somit gilt ebenfalls $w, i \models \psi_j \mathtt{U} \chi_j$.

Sei schlussendlich $\psi = \varphi \mathtt{R} \psi' \in M_i$. Wie im vorigen Fall gilt aufgrund der Definition einer Hintikka-Menge wiederum, dass $\psi' \in M_i$ und entweder $\varphi \in M_i$ oder $\bigcirc(\varphi \mathtt{R} \psi') \in M_i$. Letzteres bedeutet natürlich einfach nur $\psi \in M_{i+1}$. Dies lässt sich ebenfalls iterieren. Entweder es gilt, dass $\psi' \in M_{i'}$ für alle $i' \geq i$, oder es gibt es $h \geq i$, so dass $\{\varphi, \psi'\} \subseteq M_h$ und $\psi' \in M_{i'}$ für $i \leq i < h$. Jetzt lässt sich die Induktionshypothese in diesen Fällen auf $\varphi$ und $\psi'$ anwenden, was je nach Fall eine Aussage darüber liefert, an welchen Stellen diese Formeln im Modell gelten. In beiden Fällen folgt jedoch, dass dann auch $w, i \models \varphi \mathtt{R} \psi'$ gelten muss. □

Fasst man Lemma 11.6, Satz 11.12 und Satz 11.9 zusammen, so ergibt sich folgendes.

**Korollar 11.13.** *Für jede LTL-Formel $\varphi$ existiert ein NBA $\mathcal{A}_\varphi$, so dass $L(\mathcal{A}_\varphi) = L(\varphi)$ und $|\mathcal{A}_\varphi| = 2^{\mathcal{O}(|\varphi|)}$.*

## 11.3 LTL und alternierende Büchi-Automaten

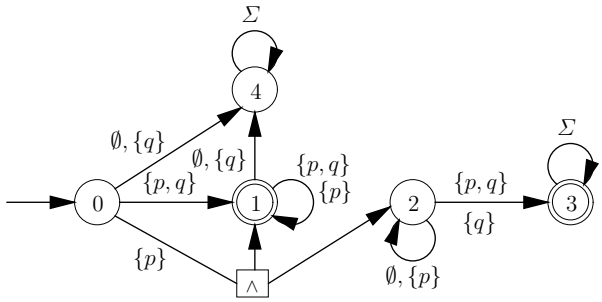
Wir wollen noch eine weitere Übersetzung von LTL in ein Automatenmodell, nämlich die alternierenden Automaten kennenlernen. Alternierende Automaten können zu konzeptuell einfacheren Übersetzungen führen, weil dieses Automatenmodell den aussagenlogischen Teil einer Logik durch das Vorhandensein von Disjunktionen und Konjunktionen in den Transitionstabellen einfacher nachbilden kann. In dem hier vorliegenden Fall von LTL bieten sich alternierende Automaten auch noch an, weil sich auf ihnen leicht eine Spezialklasse definieren lässt, die *genau* die LTL-definierbaren Sprachen erkennt. Dies ist bei NBAs nicht so einfach möglich.

Zur Erinnerung: Für eine positiv Boole'sche Formel $\delta(q, a)$ und eine Element $q'$ schreiben wir einfach $q' \in \delta(q, a)$ um anzudeuten, dass $q'$ rein syntaktisch in der Formel auftritt.

**Definition 11.14.** Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein alternierender Büchi-Automat mit $Q = \{q_0, \ldots, q_n\}$. Dieser heißt *schleifenfrei* (oder auch *zählerfrei*, bzw. *sehr schwach*) (VWABA), wenn es eine lineare Ordnung $\leq$ auf seiner Zustandsmenge gibt, so dass für alle $q, q' \in Q$ gilt: wenn $q' \in \delta(q, a)$ für ein $a \in \Sigma$, dann gilt $q' \leq q$.

Jeder VWABA ist auch ein WABA. Die geforderte Funktion $\Phi$ bildet jeden Zustand auf seine Position in der linearen Ordnung ab. Umgekehrt kann ein VWABA als WABA definiert werden, für den die Funktion $\Phi$ injektiv ist. Das bedeutet, dass jede Zustandsänderung mit einem echten Absenken des $\Phi$-Werts einhergeht.

*Beispiel 11.15.* Die Sprache aller Wörter über den Propositionen $\{p, q\}$, die an jeder Stelle $p$ und an mindestens einer Stelle $q$ erfüllen, wird durch die LTL-Formel $\mathtt{G}p \wedge \mathtt{F}q$ beschrieben. Sie wird auch von dem folgenden VWABA erkannt.




Das Symbol $\wedge$ in der verzweigenden Transition von Zustand 0 ausgehend deutet an, dass die Transitionsfunktion diesen Zustand unter dem Alphabetsymbol $\{p\}$ auf $1 \wedge 2$ abbildet.

Die lineare Ordnung auf der Zustandsmenge ist durch die (umgekehrte) Ordnung auf den Zustandsnamen gegeben und außerdem leicht aus der Graphstruktur abzulesen.

Im Gegensatz zu WABA erkennen VWABA weniger als die $\omega$-regulären Sprachen. Wir werden hier lediglich zeigen, dass VWABAs gleich mächtig zu LTL sind. Zuerst widmen wir uns der Übersetzung einer LTL-Formel in einen VWABA. Diese ist sehr simple, wenn auch vielleicht auf den ersten Blick ungewohnt. Hier wird praktisch die Formel selbst einfach als ABA aufgefasst.

**Satz 11.16.** *Für jede LTL-Formel $\theta$ über einem Alphabet $\Sigma$ existiert ein VWABA $\mathcal{A}_\theta$ mit $L(\mathcal{A}_\theta) = L(\theta)$ und $|\mathcal{A}_\theta| = \mathcal{O}(|\theta|)$.*

*Beweis.* Aufgrund von Lemma 11.6 können wir davon ausgehen, dass $\theta$ unter moderatem Blow-Up bereits in positive Normalform umgewandelt wurde. Wir definieren den ABA $\mathcal{A}_\theta$ als $(Q_\theta \cup \{\top, \bot\}, \Sigma, \theta, \delta, F)$ mit den folgenden Komponenten.

- $Q_\theta = \{\mathtt{q}_\psi \mid \psi \in \mathit{Sub}(\theta)\}$, wobei $\mathit{Sub}(\theta)$ die Menge aller Unterformeln von $\theta$ ist,
- Die Transitionsfunktion wird induktiv über den Formelaufbau für alle $a \in \Sigma$ definiert wird als

$$
\begin{aligned}
\delta(\top, a) &= \top \\
\delta(\bot, a) &= \bot \\
\delta(\mathtt{q}_p, a) &= \begin{cases} \top & \text{, falls } p \in a \\ \bot & \text{, sonst} \end{cases} \\
\delta(\mathtt{q}_{\neg p}, a) &= \begin{cases} \top & \text{, falls } p \notin a \\ \bot & \text{, sonst} \end{cases} \\
\delta(\mathtt{q}_{\varphi \vee \psi}, a) &= \delta(\mathtt{q}_\varphi, a) \vee \delta(\mathtt{q}_\psi, a) \\
\delta(\mathtt{q}_{\varphi \wedge \psi}, a) &= \delta(\mathtt{q}_\varphi, a) \wedge \delta(\mathtt{q}_\psi, a) \\
\delta(\mathtt{q}_{\bigcirc \varphi}, a) &= \mathtt{q}_\varphi \\
\delta(\mathtt{q}_{\varphi \mathtt{U} \psi}, a) &= \delta(\mathtt{q}_\psi, a) \vee (\delta(\mathtt{q}_\varphi, a) \wedge \mathtt{q}_{\varphi \mathtt{U} \psi}) \\
\delta(\mathtt{q}_{\varphi \mathtt{R} \psi}, a) &= \delta(\mathtt{q}_\psi, a) \wedge (\delta(\mathtt{q}_\varphi, a) \vee \mathtt{q}_{\varphi \mathtt{R} \psi}) \ .
\end{aligned}
$$

Man vergewissere sich, dass diese rekursive Definition wohl-definiert ist.

- Letztendlich ist $F = \{\top\} \cup \{\mathtt{q}_{\varphi \mathtt{R} \psi} \mid \varphi \mathtt{R} \psi \in \mathit{Sub}(\theta)\}$.

Die Aussage über die Größenbeschränkung von $\mathcal{A}_\theta$ ergibt sich sofort aus der Konstruktion. Es ist nicht schwer zu sehen, dass $\mathcal{A}_\theta$ tatsächlich ein VWABA ist. Als Zeuge nimmt man eine beliebige Linearisierung der Ordnung, die durch die Unterformelrelation gegeben ist und behandelt $\top, \bot$ entsprechend. Man beachte, dass $\mathcal{A}_\theta$ von einem Zustand $\mathtt{q}_\psi$ niemals einen Zustand $\mathtt{q}_{\psi'}$ erreichen kann, wenn $\psi'$ nicht Unterformel von $\psi$ ist.

Der Beweis der Korrektheit verbleibt als Übung. □

**Satz 11.17.** *Für jeden VWABA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ über einem Alphabet $\Sigma = 2^{\mathcal{P}}$ existiert eine LTL-Formel $\varphi_\mathcal{A}$ mit $L(\varphi_\mathcal{A}) = L(\mathcal{A})$ und $|\varphi_\mathcal{A}| = \mathcal{O}(|Q| \cdot 2^{|\Sigma| \cdot m})$, wobei $m := \max\{|\delta(q, a)| \mid q \in Q, a \in \Sigma\}$.*

*Beweis.* Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein VWABA mit $Q = \{q_0, \ldots, q_n\}$. O.B.d.A. nehmen wir an, dass $q_0 > q_1 > \ldots > q_n$ gilt, d.h. jeder Zustand ist vom Anfangszustand aus erreichbar.

Wir definieren nun für $i = n, \ldots, 0$ Formeln $\psi_i$, die genau diejenigen Sprachen definieren, die von $\mathcal{A}$ in Zustand $q_i$ erkannt werden. Bei der Definition von $\psi_i$ können wir davon ausgehen, dass $\psi_j$ für alle $j > i$ bereits definiert ist. Die Konstruktion von $\psi_i$ unterscheidet zwei Fälle.

Fall 1, sei $q_i \notin F$. Die von Zustand $q_i$ aus erkannte Sprache ist die kleinste Menge $X$, die die Gleichung

$$X = \bigcup_{a \in \Sigma} \{aw \mid w \in (\delta(q_i, a))'\}$$

erfüllt, wobei

$$\begin{aligned}(\varphi_1 \vee \varphi_2)' &:= \varphi_1' \cup \varphi_2' \\ (\varphi_1 \wedge \varphi_2)' &:= \varphi_1' \cap \varphi_2' \\ q_i' &:= X \\ q_j' &:= L(\psi_j) \ , \quad \text{falls } j > i\end{aligned}$$

Intuitiv besagt diese Gleichung, dass zu $X$ alle Wörter gehören, die mit einem beliebigen Alphabetsymbol $a$ beginnen, so dass das erste Suffix die Transition unter diesem Buchstaben $a$ im Zustand $q_i$ erfüllt. Es handelt sich um die kleinste Lösung, da $q_i \notin F$. Jeder Pfad in einem akzeptierenden Lauf, der den Zustand $q_i$ besucht, muss diesen auch irgendwann verlassen. Man kann sich dies so vorstellen, dass man, um die Lösung zu berechnen, erst einmal $X$ als $\emptyset$ annimmt. Dies entspricht der Menge aller Wörter, die von $q_i$ aus erkannt werden, ohne $q_i$ überhaupt zu durchlaufen. Dies ist natürlich kein einziges Wort. Die Menge derjenigen Wörter, die von $q_i$ aus erkannt werden, wobei dieser Zustand höchstens einmal durchlaufen wird, erhält man, in dem man $\emptyset$ für $X$ einsetzt und laut dieser Gleichung einen neuen Wert für $X$ berechnet. Allgemein erhält man so durch Iterieren die Menge aller Wörter, die von $q_i$ aus erkannt werden, indem dieser Zustand höchstens 0-, 1-, 2-, ... -mal durchlaufen wird. Da die Gleichung monoton ist, erhält man so immer größere Sprachen bzgl. $\subseteq$. Das Verfahren konvergiert gegen die sogenannte kleinste Lösung bzw. den kleinsten Fixpunkt dieser Gleichung.

Der Trick besteht nun darin zu erkennen, dass dieser kleinste Fixpunkt in LTL mithilfe des `U`-Operators ausdrückbar ist. Zunächst lässt sich die obige Gleichung direkt in LTL übersetzen. Hier steht $X$ nun als Variable für eine Formel statt für eine Sprache. Lösung ist also eine Formel $\theta$, so dass $L(\theta)$ genau der oben diskutierten kleinsten Lösung entspricht. Die Gleichung lautet

$$X \equiv \bigvee_{a \in \Sigma} \chi_a \wedge \bigcirc(\delta(q_i, a))'$$

wobei $\chi_a$ wiederum die charakteristische Formel für das Alphabetsymbol $a$ ist und

$$\begin{aligned}(\varphi_1 \vee \varphi_2)' &:= \varphi_1' \vee \varphi_2' \\ (\varphi_1 \wedge \varphi_2)' &:= \varphi_1' \wedge \varphi_2' \\ q_i' &:= X \\ q_j' &:= \psi_j \ , \quad \text{falls } j > i\end{aligned}$$

Mithilfe der LTL-Äquivalenzen $\bigcirc(\varphi_1 \wedge \varphi_2) \equiv \bigcirc\varphi_2 \wedge \bigcirc\varphi_2$, $\bigcirc(\varphi_1 \vee \varphi_2) \equiv \bigcirc\varphi_2 \vee \bigcirc\varphi_2$ und der deMorgan'schen Regeln lässt sich die rechte Seite der Gleichung in disjunktive Normalform bringen, so dass $\bigcirc$-Operatoren nur direkt vor atomaren Formeln vorkommen. Die rechte Seite ist also äquivalent zu einem $\alpha \vee (\beta \wedge \bigcirc X)$, so dass $X$ nicht in $\alpha$ oder $\beta$ vorkommt. Damit gilt dann $X \equiv \beta \mathtt{U} \alpha$ und wir können $\psi_i$ somit als $\alpha \mathtt{U} \beta$ definieren.

Fall 2, sei $q_i \in F$. Ebenso lässt sich hier eine Gleichung $X \equiv \alpha \wedge (\beta \vee \bigcirc X)$ durch Umformen in konjunktive Normalform finden, die genau die von $q_i$ aus erkannte Sprache beschreibt. Hierbei handelt es sich jedoch um den größten Fixpunkt, da von $q_i$ aus auch Wörter akzeptiert werden, deren Läufe unendlich lange in $q_i$ verweilen können. Dann gilt aber auch $X \equiv \mathtt{G}\alpha \vee (\alpha \mathtt{U} (\beta \wedge \alpha))$, was wir als $\psi_i$ hernehmen können.

Die Größenbeschränkung ergibt sich aus der Tatsache, dass es insgesamt $n$ viele Formeln der Form $\psi_i$ gibt und jede aus einer Formel der Größe $\mathcal{O}(|\Sigma| \cdot m)$ durch Umwandeln in dis-/konjunktive Normalform entsteht, wobei $m$ eine obere Schranke an die Größe einer Boole'schen Formel in der Transitionstabelle des VWABA ist. □

Die Übersetzung von LTL nach VWABA ist also linear, die von VWABA nach LTL in der hier präsentierten Fassung einfach exponentiell in der Größe des Automaten, aber doppelt exponentiell in der Anzahl der verwendeten Propositionen. Dies ist nicht optimal. Der Grund für den zweifachen Exponenten liegt in der suboptimalen Verwendung von Alphabetsymbolen. Man kann alternierende Automaten über Alphabeten der Form $\Sigma = 2^{\mathcal{P}}$ auch mit einer Transitionsfunktion der Form $Q \to \mathcal{B}^+(Q \cup \mathcal{P})$ im Gegensatz zu dem hier verwendeten $Q \times 2^{\mathcal{P}} \to \mathcal{B}^+(Q)$ definieren, denn es gilt folgendes.

$$Q \times 2^{\mathcal{P}} \to \mathcal{B}^+(Q) \;\simeq\; Q \to (2^{\mathcal{P}} \to \mathcal{B}^+(Q))$$

Außerdem sind positiv Boole'sche Formel ausdrucksstark genug, so dass man Funktionen vom Typ $2^{\mathcal{P}} \to \mathcal{B}^+(Q)$ darin ausdrücken kann. Somit lässt sich mit Transitionsfunktionen vom Typ $Q \to \mathcal{B}^+(Q \cup \mathcal{P})$ mindestens soviel ausdrücken wie mit den hier verwendeten. Intuitiv wird dort also jedem Zustand eine Kombination aus Propositionen im nächstgelesenen Alphabetsymbol und Nachfolgezuständen zugeordnet. Insbesondere lässt sich damit der doppelt exponentielle Blow-Up zu einem einfach exponentiellen reduzieren.

## 11.4 Spezifikation und Verifikation

Eine wichtige Anwendung der Logik LTL ist ihre Verwendung als Spezifikationssprache für reaktive und parallele Systeme. An dieser Stelle wollen wir weniger auf Fragen der Modellierung solcher Systeme eingehen, sondern uns auf den Einsatz von LTL und—damit verbunden—automatentheoretischen Methoden konzentrieren. Systeme können dabei Software oder Hardware sein; "parallel" bedeutet, dass diese typischerweise aus mehreren Komponenten bestehen, die miteinander interagieren; "reaktiv" soll hier nur bedeuten, dass diese nicht unbedingt terminieren müssen. Ein Beispiel dafür sind z.B. Betriebssysteme. In dem im Folgenden vorgestellten Szenario geht es hauptsächlich darum, das nach außen hin sichtbare Verhalten solch eines Systems in Anlehnung an eine Spezifikation zu verifizieren. Weniger wichtig in dieser Modellierungssicht sind also z.B. Fragen nach der Korrektheit der Berechnungen

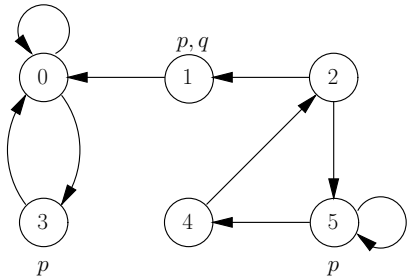



**Abb. 11.1.** Beispiel für ein Transitionssystem.

von internen Methoden des Systems. Typische Fragestellungen sind z.B. "wird jede Anfrage irgendwann später beantwortet" etc.

Wir werden zunächst ein sehr intuitives und vereinfachendes Modell für solche Systeme vorstellen, die sogenannten Transitionssysteme. Diese kann man einfach als Rahmenwerk für die operationale Semantik eines Programms ansehen. Danach wenden wir uns der Frage zu, wie man (un-)erwünschtes Verhalten solch eines Systems in LTL spezifiziert und verifiziert.

### 11.4.1 Transitionssysteme und Läufe

**Definition 11.18.** Sei $\mathcal{P}$ eine endliche Menge atomarer Propositionen. Ein *Transitionssystem* ist ein Tupel $\mathcal{T} = (S, \longrightarrow, I, \lambda)$, wobei $(S, \longrightarrow)$ ein gerichteter Graph mit Knotenmenge $S$ und Kantenrelation $\longrightarrow$, $I \subseteq S$ eine ausgezeichnete Menge von *Anfangszuständen* und $\lambda : S \to 2^{\mathcal{P}}$ ist.

Ein solches Transitionssystem ist *total*, falls es für alle $s \in S$ ein $t \in S$ gibt mit $s \longrightarrow t$.

Ein vernünftiges Maß für die Größe eines totalen Transitionssysteme ist das folgende: $|\mathcal{T}| := | \longrightarrow | + |I|$.

Im folgenden beschränken wir uns auf totale Transitionssysteme, ohne dies jedes Mal explizit anzumerken.

Ein Transitionssystem repräsentiert in natürlicher Weise das zeitliche Verhalten eines zustandsbasierten Programms. Ein Beispiel für ein Transitionssystem ist in Abb. 11.1 gezeigt. Dies repräsentiert ein Programm, welches 6 verschiedene Zustände annehmen kann. Übergänge sind nach der Kantenrelation in dem gerichteten Graphen möglich. Man beachte, dass das Programm nichtdeterministisch ist. In dem Zustand 1 gelten z.B. die atomaren Aussagen $p$ und $q$, während in 2 z.B. beide nicht gelten, während z.B. im Zustand 3 nur die Aussage $p$ wahr ist.

Man kann solch ein Transitionssystem als (evtl. endliche) Repräsentation von (evtl. unendlich) vielen Abläufen des zugrundeliegenden Programms ansehen. Der Transitionsgraph stellt also evtl. Nichtdeterminismus dar, der

z.B. durch nichtdeterministische Auswahl im Fortschreiten einzelnen Komponenten im Gesamtsystem entsteht. Ein Lauf dagegen ist deterministisch in dem Sinn, dass zu jedem Zustand genau ein Folgezustand gehört. Ein Lauf entspricht einer möglichen Art des Gesamtsystems, sich konkret zu verhalten.

**Definition 11.19.** Sei $\mathcal{T} = (S, \longrightarrow, I, \lambda)$ ein totales Transitionssystem über atomaren Propositionen $\mathcal{P}$. Ein *Pfad* (in $\mathcal{T}$) ist eine unendliche Sequenz $s_0 s_1 \ldots \in S^\omega$, so dass $s_0 \in I$ und $s_i \longrightarrow s_{i+1}$ für alle $i \in \mathbb{N}$.

Ein *Lauf* (in $\mathcal{T}$) ist ein unendliches Wort $a_0 a_1 \ldots \in (2^\mathcal{P})^\omega$, so dass es einen Pfad $s_0 s_1 \ldots$ gibt mit $a_i = \lambda(s_i)$ für alle $i \in \mathbb{N}$. Wir schreiben $L(\mathcal{T})$ für die Menge aller Läufe in $\mathcal{T}$.

Ein Lauf in einem Transitionssystem entsteht also durch Ablesen der Knotenbeschriftungen entlang eines unendlichen Pfades. Die Totalität garantiert, dass jeder maximale Pfad unendlich lang ist.

### 11.4.2 Model-Checking

Das Model-Checking-Problem für LTL besteht nun darin, zu einem gegebenen, totalen Transitionssystem $\mathcal{T}$ und einer gegebenen LTL-Formel $\varphi$ zu entscheiden, ob $L(\mathcal{T}) \subseteq L(\varphi)$ gilt. Dies ist die Formalisierung des folgenden Problems: Gegeben ist eine Implementierung eines Systems ($\mathcal{T}$) sowie eine Spezifikation darüber, wie sich das System verhalten soll ($\varphi$). Es soll nun herausgefunden werden, ob das System korrekt in Bezug auf die Spezifikation ist oder nicht. Im Rahmen von LTL—und allen anderen Formalismen, die Mengen unendlicher Wörter definieren—bedeutet Korrektheit in diesem Kontext, dass alle möglichen Läufe des Systems durch die Spezifikation abgedeckt sind, d.h. es gibt keinen Lauf im System, der nicht in der Sprache der Spezifikation ist.

Im Folgenden zeigen wir, wie mithilfe der Übersetzung von LTL in NBAs das Model-Checking-Problem gelöst werden kann. Intuitiv betrachtet man das Transitionssystem als NBA, der genau die Sprache aller Läufe erkennt. Dann konstruiert man das Produkt aus diesem und einem NBA, der das *Komplement* der Spezifikation erkennt und testet dieses auf Leerheit, denn es gilt $A \subseteq B \iff A \cap \overline{B} = \emptyset$.

**Definition 11.20.** Sei $\mathcal{T} = (S, \longrightarrow, I, \lambda)$ ein Transitionssystem über $\mathcal{P}$ und $\mathcal{A} = (Q, 2^\mathcal{P}, q_0, \delta, F)$ ein NBA. Wir definieren deren *Produkt* als NBA $\mathcal{T} \otimes \mathcal{A} := (S \times Q \cup \{\mathsf{q}_{\mathsf{init}}\}, \{\bullet\}, \mathsf{q}_{\mathsf{init}}, \Delta, S \times F)$, mit

$$\begin{aligned} \Delta\big((s,q),\bullet\big) &:= \{(s',q') \mid s \longrightarrow s' \text{ und } q' \in \delta(q,\lambda(s))\} \\ \Delta\big(\mathsf{q}_{\mathsf{init}},\bullet\big) &:= \{(s',q') \mid \exists s \in I . s \longrightarrow s' \text{ und } q' \in \delta(q_0,\lambda(s))\} \end{aligned}$$

Der ausgezeichnete Anfangszustand $\mathsf{q}_{\mathsf{init}}$ wird hier nur eingeführt, damit der resultierende NBA einen einzigen Anfangszustand hat. Da ein Transitionssystem jedoch mehrere haben kann, hätte auch intuitiv das Produkt aus diesem und einem NBA mehrere Anfangszustände. Der Zustand $\mathsf{q}_{\mathsf{init}}$ wird

dann so benutzt, dass er das Verhalten aller Paare aus Anfangszuständen der beiden Komponenten modelliert.

**Lemma 11.21.** *Sei $\mathcal{T}$ ein Transitionssystem und $\mathcal{A}$ ein NBA. Dann gilt $L(\mathcal{T} \otimes \mathcal{A}) = \emptyset$ genau dann, wenn $L(\mathcal{T}) \cap L(\mathcal{A}) = \emptyset$.*

*Beweis.* Übung.

Dies liefert einen algorithmischen Zugang zum Model-Checking-Problem. Ist die Spezifikation als NBA gegeben, so muss dieser komplementiert werden, und es wird getestet, ob das Produkt aus diesem und der Spezifikation *nicht* das Wort $\bullet^\omega$ erkennt. Komplementierung von NBA ist—wie oben gesehen—nicht sehr einfach. Ist die Spezifikation jedoch als LTL-Formel gegeben, so ist dieser Schritt sehr einfach, da Negation in der Logik vorhanden ist.

**Satz 11.22.** *Das Model-Checking-Problem mit Transitionssystem $\mathcal{T}$ und LTL-Formel $\varphi$ lässt sich in Zeit $|\mathcal{T}| \cdot 2^{\mathcal{O}(|\varphi|)}$ entscheiden.*

*Beweis.* Seien $\mathcal{T}$ und $\varphi$ gegeben. Betrachte dann $\neg\varphi$. Nach Korollar 11.13 gibt es einen NBA $\mathcal{A}_{\neg\varphi}$ mit $L(\mathcal{A}_{\neg\varphi}) = \overline{L(\mathcal{A})}$ und $|\mathcal{A}_{\neg\varphi}| = 2^{\mathcal{O}(|\varphi|)}$. Somit gilt $|\mathcal{T} \otimes \mathcal{A}_{\neg\varphi}| = |\mathcal{T}| \cdot 2^{\mathcal{O}(|\varphi|)}$ und laut Lemma 11.21 gilt $L(\mathcal{T} \otimes \mathcal{A}_{\neg\varphi}) \neq \emptyset \iff L(\mathcal{T}) \cap L(\mathcal{A}_{\neg\varphi}) \neq \emptyset$. Somit reduziert sich das Model-Checking-Problem auf das Leerheitsproblem für NBA. Dieses ist laut Korollar 9.5 jedoch in linearer Zeit lösbar. □

# Notizen

Die Automatentheorie auf unendlichen Wörtern geht hauptsächlich auf die Arbeit J. Büchis von 1962 zurück [Büc62], in der er endliche Automaten auf unendlichen Wörtern einsetzte, um ein Entscheidungsverfahren für MSO zu gewinnen. Dies betrifft insbesondere die hier vorgestellten Sätze 5.13 zur Charakterisierung der $\omega$-regulären Sprachen durch das Modell des Büchi-Automaten, 6.8 zum Komplementabschluss dieser Klasse und 6.11 zur Entscheidbarkeit der MSO. Wie hier auch deutlich wird, ist der Komplementabschluss das größte Problem im Entscheidbarkeitsbeweis. Dieses Problem hat noch weiter Aufmerksamkeit auf sich gezogen; so entstanden weitere Beweise für diese Abschlusseigenschaft, u.a. von N. Klarlund [Kla91], A. Sistla, M. Vardi und P. Wolper [SVW87], und die hier in Satz 10.16 und dann in Kor. 10.17 ebenfalls präsentierte Technik mittels alternierender Automaten von O. Kupferman und M. Vardi [KV01].

Das Lemma von König (Satz 6.1) ist ein klassisches Resultat in der Kombinatorik, welches in einer Arbeit von D. König gezeigt wurde [Kön27]. Ähnlich alt ist der ebenfalls berühmte Satz von F. Ramsey [Ram30].

Da die Komplementierung für NBA in der Praxis ungleich schwieriger als die für NFA ist, können wir hier auch kein Tool wie MONA für MSO vorgestellen. Dass es nichts vergleichbares gibt, liegt u.a. daran, dass viele Verfahren zur Komplementierung von NBA nicht symbolisch mithilfe von BDDs [Bry86] implementiert werden können, so wie dies bei den NFAs in MONA gemacht wird, und dass es auch nicht so gute Minimierungsverfahren für NBAs gibt. Letzteres liegt daran, dass das Finden eines minimalen, zu einem gegebenen äquivalenten NBA komplexitätstheoretisch schwierig ist. Deswegen geben existierende Verfahren, wie z.B. das von S. Juvekaar und N. Piterman [JP06] den Anspruch der Minimalität auf und stellen somit eigentlich nur Heuristiken dar. Üblicherweise basieren diese Verfahren dann auf der Berechnung von Simulationsrelationen.

Die Erfinder der meisten anderen Automatentypen auf unendlichen Wörtern sind in deren Namen verewigt: Rabin-Automaten sind nach M. O. Rabin benannt, Streett-Automaten nach R. S. Streett und Muller-Automaten nach

D. E. Muller [Mul63]. Die Paritätsbedingung bildet dabei eine Ausnahme—sie wurde von A. Mostowski erfunden [Mos84].

Die Safra-Konstruktion wurde 1988 von S. Safra in einer viel beachteten Arbeit vorgestellt [Saf88] und ist auch in seiner Dissertation schön aufbereitet [Saf89]. Der Grund dafür, dass seine Arbeit soviel Anklang gefunden hat, ist der, dass sie eine verwendbare Konstruktion liefert. R. McNaughton hatte 1966 bereits zeigen können, dass jede $\omega$-reguläre Sprachen von einem deterministischen Muller-Automaten erkannt wird [McN66]. Sein Beweis liefert jedoch kein Verfahren, um den DMA aus einem gegebenen NBA zu konstruieren. Dieses Resultat ist auch noch auf andere Weise bewiesen worden, z.B. von W. Thomas aus dem Jahre 1981 [Tho81]; aber erst durch Safras Beweis lässt sich der DMA explizit und effizient konstruieren.

Mit der Zeit hat man jedoch gemerkt, dass auch Safras Determinisierungsprozedur für viele Fälle nicht besonders geeignet ist. Dies betrifft Fälle, in denen ein NBA nicht nur determinisiert, sondern auch komplementiert werden muss. Dafür hat N. Piterman wie hier vorgestellt die Safra-Konstruktion dahingehend verfeinert, dass sie direkt einen leicht zu komplementierenden Paritätsautomaten liefert [Pit06]. Ähnliche Verfeinerungen stammen von D. Kähler und T. Wilke [KW08] und S. Schewe [Sch09]. Die hier ebenfalls vorgestelle Methode, die den Umweg über Muller-Automaten mithilfe der Latest Appearance Records macht, geht auf Y. Gurevich und L. Harrington zurück [GH82].

Da Determinisierung von NBAs zumindest für die Praxis weiterhin nicht zufriedenstellend gelöst ist, sind z.B. O. Kupferman und M. Vardi dazu übergegangen, die Determinisierung in gewissen Anwendungsfällen, insbesondere bei logischen Entscheidungsverfahren, zu umgehen [KV05].

Die oberen Schranken, die man für die Komplementierung einer $\omega$-regulären Sprache über die hier vorgestellten Determinisierungsprozeduren erhält, sind fast optimal. Die in Bsp. 8.15, Satz 8.16 und Kor. 8.17 gegebenen unteren Schranken für die Komplementierung und Determinisierung von NBA wurden von M. Michel [Mic88] gefunden. Detailliert aufgeschrieben ist der Beweis bei W. Thomas [Tho97], siehe auch C. Lödings Arbeit [Löd99].

Satz 9.3 war lange Zeit unbewiesen, d.h. es war nicht bekannt, wie man die Zerlegung eines Graphen in starke Zusammenhangskomponenten in linearer Zeit berechnen könnte. Erst E. Tarjan hat 1972 ein Verfahren vorgestellt, welches auf zwei aufeinanderfolgenden Tiefensuchen beruht und dieses Problem löst [Tar72].

Das Ramsey-basierte Entscheidungsverfahren für die Universalität von NBA wurde zunächst von C. Lee, A. Ben-Amram und N. Jones für den Spezialfall der Size-change Terminationsanalyse verwendet [LJBA01]; allerdings enthält auch Safras Dissertation ähnliche Ideen im Zusammenhang mit Streett-Automaten. Die allgemeine Verwendung für das NBA-Universalitätsproblems wurde von S. Fogarty und M. Vardi beschrieben [FV10]. Die Methode von Jones et al. lässt sich auch auf Paritätsautomaten anwenden [DHL06].

Das Konzept der Alternierung wurde im Rahmen allgemeiner Turing-Maschinen und der Komplexitätstheorie von A. Chandra, D. Kozen und L. Stockmeyer erfunden [CKS81]. Alternierende, endliche Automaten wurden dann insbesondere von D. Muller und P. Schupp einerseits [MS87] und von M. Vardi und O. Kupferman andererseits untersucht [KV01]. Von diesen stammt auch die hier präsentierte Analyse über gedächtnislose Läufe eines ABA. Die dort verwendeten Techniken wurden von ihnen außerdem auch schon früher benutzt, um eine Verbindung zwischen alternierenden Automaten auf Wörtern und nichtdeterministischen Baumautomaten, um die es in den nächsten zwei Teilen des Buchs geht, darzulegen [KV98].

Die Konstruktion im Beweis von Satz 10.7, die einen ABA in einen NBA übersetzt, wurde 1984 von S. Miyano und T. Hayashi vorgestellt [MH84] und ist deswegen auch als Miyano-Hayashi-Konstruktion bekannt.

Die hier vorgestellte Komplementierung eines NBA mittels Alternierung von O. Kupferman und M. Vardi liefert zwar eine asymptotisch nicht optimale obere Schranke an die Größe der resultierenden Automaten. Sie hat jedoch den Vorteil, dass sie leichter symbolisch implementiert werden kann [KC09], da die Zustände der resultierenden Automaten nur aus Mengen von Paaren bestehen, während die bekannten Verfahren, die auf Determinisierung beruhen, mit Safra-Bäumen arbeiten, die sich nicht so leicht mittels BDDs repräsentieren lassen.

Schwache Automaten sind eine Erfindung von D. Muller, A. Saoudi und P. Schupp [MSS88]; wobei diese allerdings die folgende offensichtlich äquivalente Definition zugrunde legen: Die Funktion $\Phi : Q \to \{0, \ldots, n-1\}$ muss so beschaffen sein, dass wenn $\Phi(q) = \Phi(q')$, dann $q \in F \iff q' \in F$ gilt. schleifenfreie, alternierende Automaten wurden von C. Löding und W. Thomas definiert [LT00]. Aus dieser Arbeit stammt auch die hier präsentierte Übersetzung eines VWABA in eine LTL-Formel.

In diesem Buch haben wir erststufige Logik nicht explizit auf unendlichen Wörtern betrachtet. Man kann jedoch genauso wie auf endlichen Wörtern die Klasse der FO-definierbaren Sprachen algebraisch als die stern-freien Sprachen charakterisieren. Dies wurde von W. Thomas gemacht [Tho79]. Eine interessante Frage in diesem Zusammenhang ist, wie sich LTL und FO zueinander verhalten. Der Beweis, dass LTL-definierbare Sprachen auch FO-definierbar sind, ist trivial. Umso schwerer ist es, die Rückrichtung zu zeigen. Dies wird oft als Kamp's Theorem bezeichnet, zu finden in dessen Dissertation [Kam68]; jedoch hat H. Kamp "nur" gezeigt, dass FO über gewissen, linearen Strukturen in LTL mit Vergangenheitsoperatoren übersetzt werden kann. Dies lässt sich mit D. Gabbay's Separation Theorem [Gab89] kombinieren, welches dann jede Formel dieser erweiterten Logik in eine normale LTL-Formel überführt. Einen direkten Beweis der LTL-Definierbarbeit von FO-definierbaren Sprachen findet man bei D. Gabbay, A. Pnueli, S. Shelah und J. Stavi [GPSS80]. Einen guten Überblick über FO-Definierbarkeit auf endlichen und unendlichen Wörtern sowie alternativen Charakterisierungen findet man bei V. Diekert und P. Gastin [DG08].

Die Fixpunkttheorie hinter den temporalen Operatoren ist hier nur am Rande, genauer gesagt bei der Argumentation im Beweis von Satz 11.17 angerissen worden. Einen besseren Überblick über die Zusammenhänge zwischen Fixpunkten monotoner Gleichungen und Iterationsverfahren zum Bestimmen dieser Fixpunkte bieten verschiedene Textbücher [DP02, Win93]. Die Theorie von Fixpunkten monotoner Gleichungen geht auf fundamentale Arbeiten von B. Knaster und A. Tarski zurück [Kna28, Tar55].

Der Zusammenhang zwischen LTL und NBA wurde zuerst von P. Sistla, M. Vardi und P. Wolper [SVW83, VW94] beschrieben. Die hier präsentierte Übersetzung ist zwar konzeptuell simpel, aber in der Praxis nicht besonders gut. Aus diesem Grund hat es eine Vielzahl von weiteren Arbeiten zur Übersetzung von LTL-Formeln in Büchi-Automaten gegeben, so z.B. von F. Somenzi und R. Bloem [SB00], P. Gastin und D. Oddoux [GO01], D. Giannakopoulou und F. Lerda [GL02], etc.

Das Thema der Programmverifikation mittels Model-Checking wird in zwei Büchern von E. Clarke, O. Grumberg und D. Peled [CGP99] sowie von C. Baier und J.-P. Katoen [BK08] sehr vertieft behandelt. Letzteres enthält auch viele brauchbare Übungsaufgaben zu diesem Thema.

Als generelle Literatur zu den Themen, die in diesem Teil des Buchs behandelt werden, empfehlen sich die Handbuch-Artikel von W. Thomas [Tho90, Tho97] und zwei Sammelbände: ein von E. Grädel, W. Thomas und T. Wilke ediertes mit viel grundlegendem Material aus dem Jahr 2002 [GTW02], sowie die von J. Flum, E. Grädel und T. Wilke herausgegebene Festschrift zu Ehren W. Thomas' aus dem Jahr 2007 mit viel Übersichtsmaterial [FGW07].