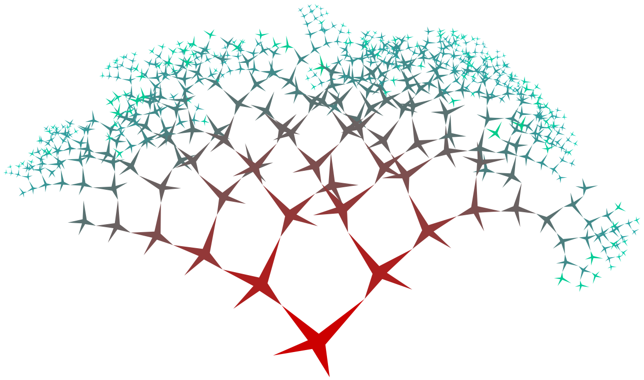
Syntaktische Methoden der Bilderzeugung

Wintersemester 1999/2000

Frank Drewes und Hans-Jörg Kreowski

(unvollständige Fassung)



Inhaltsverzeichnis

Kapitel 1

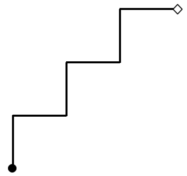
Einleitung

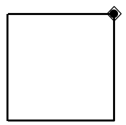
<< fehlt noch >>

Kapitel 2

Kettencode-Bildsprachen

Eine bemerkenswert einfache Möglichkeit, Bilder mit syntaktischen Methoden zu erzeugen, ergibt sich durch graphische Interpretationen von Zeichenketten. Ist beispielsweise eine Zeichenkette nur aus den vier Zeichen o, s, w und n zusammengesetzt, dann kann man sie beim Lesen von links nach rechts als Anweisung an ein Zeichengerät mit Zeichenstift (Plotter) deuten: Für jedes gelesene Zeichen wird mit dem Zeichenstift von seiner aktuellen Position aus eine gerade Linie von einer eingestellten Einheitslänge in die durch das Zeichen repräsentierte Haupthimmelsrichtung gezogen, nämlich beim Zeichen o nach Osten, bei s nach Süden, bei w nach Westen und bei n nach Norden. Der Einfachheit halber sollten die Himmelsrichtungen für das Gerät festgelegt sein, so daß bei einem Blatt Papier beispielsweise oben im Norden ist und rechts im Osten; dann kann man sich einen Kompaß sparen. Die Ketten *nonono* und *sswwnnoo* ergeben eine Treppe mit drei Stufen beziehungsweise ein Quadrat mit doppelter Einheitslänge als Seite:





Dabei ist die Einheitslänge als 1 cm gewählt, der Startpunkt der Zeichnung etwas verdickt und der Endpunkt durch ein Karo markiert.

Auf der Grundlage dieser graphischen Interpretation spezifiziert also eine Menge von Zeichenketten über dem Alphabet $\{o, s, w, n\}$ eine Menge von Zeichnungen, die eindeutig durch die Wahl eines Startpunktes, der Einheitslänge und dem Koordinatensysten festgelegt sind. Diese Beobachtung erlaubt, die Theorie formaler Sprachen auf die Erzeugung von Bildern und Bildklassen anzuwenden. So stellt jede Chomsky-Grammatik mit dem terminalen Alphabet $\{o, s, w, n\}$ durch ihre erzeugte Sprache, die ja aus Zeichenketten über dem terminalen Alphabet besteht, eine Bildsprache dar. Genau so gut kann auch jeder andere Mechanismus herangezogen werden, der Sprachen über dem Alphabet $\{o, s, w, n\}$ spezifiziert.

Die Ideen, Zeichenketten in der beschriebenen Weise zur Bilderzeugung zu nutzen, geht auf Freeman [Fre61, Fre74] zurück. Wie man formale Sprachen dafür nutzt, wird in Arbeiten von Maurer, Rozenberg und Welzl [MRW82] sowie Dassow und Hinz [DH89] ausführlich dargestellt. Insbesondere in der letzten Arbeit finden sich auch weiterführende Literaturhinweise.

Die vorausgegangenen Überlegungen sind in der folgenden Definition zusammengefaßt und formalisiert.

2.1 Definition

1. Das kartesische Produkt $\mathbb{Z}^2$, also die Menge aller Punkte in der Ebene mit ganzzahligen Koordinaten, wird als *universelle Punktmenge* bezeichnet.
2. Jeder Punkt (x, y) hat vier direkte *Nachbarn*, den *östlichen* $o(x, y) = (x + 1, y)$, den *südlichen* $s(x, y) = (x, y - 1)$, den *westlichen* $w(x, y) = (x - 1, y)$ und den *nördlichen* $n(x, y) = (x, y + 1)$.
3. Die Menge der geraden Linien zwischen allen Punkten und jeweils ihren direkten Nachbarn bildet die *universelle Menge der Einheitslinien*, die eindeutig repräsentiert werden kann durch die Menge
$$UL = \{\{(x, y), (x', y')\} \mid x, y, x', y' \in \mathbb{Z}, |x' - x| + |y' - y| = 1\}$$
aller Paare von Endpunkten von Einheitslinien.
4. Ein (fixiertes Einheitslinien-)*Bild* p ist eine endliche Teilmenge von UL.
5. Für ein Bild p ist
$$point(p) = \bigcup_{l \in p} l$$
die *Punktmenge* von p.
6. Ein Bild p heißt *zusammenhängend*, wenn für je zwei Punkte $(x, y), (x', y') \in point(p)$ eine Folge von Punkten (x_i, y_i) für $i = 0, \ldots, n$ $(n \geq 0)$ existiert mit $(x, y) = (x_0, y_0)$, $(x', y') = (x_n, y_n)$ und $\{(x_i, y_i), (x_{i+1}, y_{i+1})\} \in p$ für $i = 0, \ldots, n - 1$.
7. Ein *gezeichnetes Bild* ist ein Tripel $d = (p, b, e)$, wobei p ein zusammenhängendes Bild ist und b und e zwei Punkte sind, für die $b = e$ gilt, falls $p = \emptyset$, und $b, e \in point(p)$ sonst. Die Komponente p wird das *unterliegende Bild* genannt, b der *Anfangspunkt* und e der *Endpunkt*.
8. Eine Zeichenkette $u \in \{o, s, w, n\}^*$ heißt *Bildbeschreibung*.
9. Das *gezeichnete Bild* $drawing(u) = (p(u), b(u), e(u))$ einer Bildbeschreibung u ist rekursiv definiert durch:

   (i) $(p(\lambda), b(\lambda), e(\lambda)) = (\emptyset, (0, 0), (0, 0))$,

   (ii) $(p(vr), b(vr), (e(vr)) = (p(v) \cup \{\{e(v), r(e(v))\}\}, (0, 0), r(e(v)))$ für $v \in \{o, s, w, n\}^*$ und $r \in \{o, s, w, n\}$.

10. Sei $L \subseteq \{o,s,w,n\}^*$. Dann wird
$$picture(L) = \{p(u) \mid u \in L\}$$
die durch L repräsentierte *Kettencode-Bildsprache* genannt.

11. Eine Menge von Bildern B wird (*reguläre, kontextfreie, kontextsensitive, Typ-0-) Kettencode-Bildsprache* genannt, wenn es eine (reguläre, kontextfreie, kontextsensitive, Typ-0-) Sprache $L \subseteq \{o,s,w,n\}^*$ gibt mit $picture(L) = B$. $\diamond$

Beachte, daß bei gezeichneten Bildern von Kettencodes als Bildbeschreibungen der Anfangspunkt immer der Ursprung des Koordinatensystems ist. Der Endpunkt wird für die Rekursion gebraucht, weil die nächste Linie immer am vorher erreichten Endpunkt ansetzt. Entscheidend ist die Definition des Zeichnens anhand der Bildbeschreibung in Punkt 9. Die leere Bildbeschreibung liefert das leere Bild mit dem Ursprung des Koordinatensystems als Anfangs- und Endpunkt. Das Bild einer Bildbeschreibung der Form vr, die also mit einer Bildbeschreibung v beginnt und mit der Richtung r endet, entsteht dadurch, daß das Bild gemäß der Beschreibung v gezeichnet wird mit dem Ursprung des Koordinatensystems als Anfangspunkt und daß dann vom erreichten Endpunkt $e(v)$ noch eine Einheitslinie zum neuen Endpunkt $e(vr) = r(e(v))$ in die Richtung r gezeichnet wird. Die Linienmenge $p(v)$ wird dazu mit einer Menge vereinigt, die genau diese Linie als einziges Element enthält.

Der Sinn der eingeführten Begriffe erschließt sich teilweise bereits in folgender Beobachtung.

2.2 Beobachtung
Das gezeichnete Bild $(p(u), b(u), e(u))$ für $u \in \{o,s,w,n\}^*$ hat folgende Eigenschaften:

(i) $p(u)$ ist zusammenhängend,

(ii) $b(u) = (0,0)$,

(iii) entweder $p(u)) = \emptyset$ und $e(u) = (0,0)$, oder $p(u) \neq \emptyset$ und $b(u), e(u) \in point\ (p(u))$.

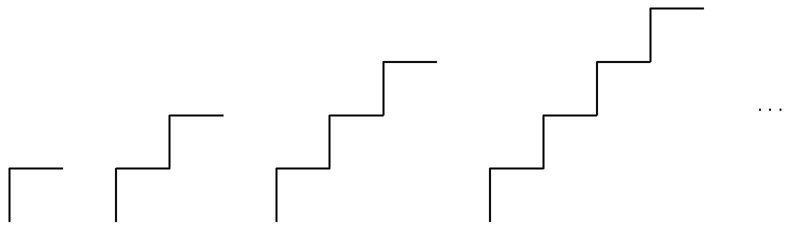
Beweis. Ist $u = \lambda$, so ist $p(u) = \emptyset$ und $b(u) = (0,0) = e(u)$. Ist $u = vr$ mit $r \in \{o,s,w,n\}$, dann kann als Induktionsvoraussetzung angenommen werden, daß $p(v)$ zusammenhängend ist, und $e(v) \in point(p(v))$. Betrachte nun $p(u) = p(vr) = p(v) \cup \{\{e(v), r(e(v))\}\}$ und $e(u) = e(vr) = r(e(v))$. Wegen
$$point(p(u)) = \bigcup_{l \in p(u)} l = \bigcup_{l \in p(v)} l \cup \{e(v), r(e(v))\}$$
ist $e(u) = r(e(v)) \in point(p(u))$. Seien nun $(x,y), (x',y') \in point(p(u))$. Dann sind im einfachsten Fall $(x,y), (x',y')$ bereits Punkte von $p(v)$, so daß beide Punkte durch Linien aus $p(v)$ verbunden sind, da $p(v)$ zusammenhängend ist. Wegen $p(v) \subseteq p(u)$ sind die beiden Punkte auch in $p(u)$ verbunden. Es bleibt der schwierigere Fall, daß $(x,y) \in point(p(v))$ und $(x',y') = r(e(v))$. Nach Voraussetzung ist $e(v) \in point(p(v))$ und $p(v)$ zusammenhängend. Deshalb gibt es eine Folge von Punkten in $p(v)$, die paarweise Linien von $p(v)$ bilden und (x,y) mit $e(v)$ verbinden. Wegen $p(v) \subseteq p(u)$ verbindet diese Folge auch die beiden Punkte in $p(u)$. Da außerdem $\{e(v), r(e(v))\}$ eine Linie von $p(u)$ ist, kann man $r(e(v))$ an die Folge

hängen, die dann (x,y) und $(x',y') = r(e(v))$ in $p(u)$ verbindet. Insgesamt erweist sich auch $p(u)$ als zusammenhängend. Schließlich gilt immer $b(u) = (0,0)$. Ist $u \neq \lambda$ und damit $p(u) \neq \emptyset$, so ist $(0,0)$ ein Endpunkt der ersten gezeichneten Linie von $p(u)$ und damit ein Element von $point(p(u))$, so daß zusammenfassend die Behauptung gezeigt ist. $\blacksquare$

2.1 Beispiele

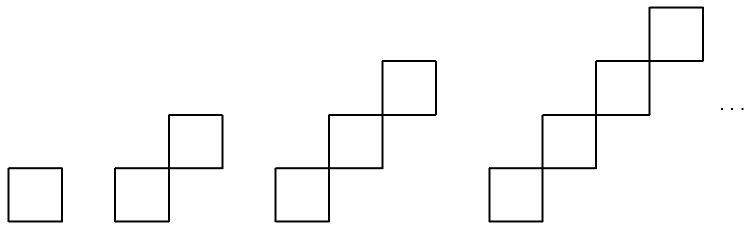
Einige Beispiele sollen das Prinzip der Kettencode-Bildsprachen illustrieren.

1. Die Sprache $L_{step} = \{(no)^k \mid k \geq 1\}$ beschreibt alle Treppen



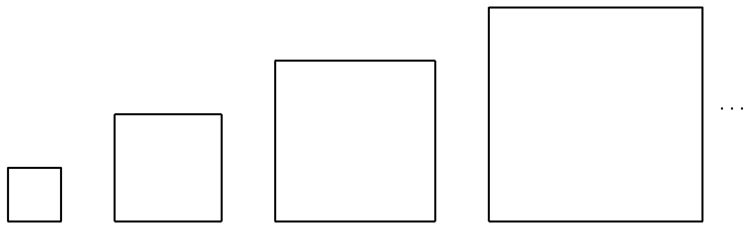
Die Bildsprache aller Treppen ist also offenbar regulär.

2. Die Sprache $L_{diagonal} = \{(no)^k(sw)^k \mid k \geq 1\}$ repräsentiert folgende Bilder:



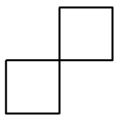
$L_{diagonal}$ kann durch folgende Regeln erzeugt werden: $S ::= noSsw \mid nosw$. Die zugehörige Bildsprache ist also kontextfrei.

3. Die Sprache $L_{square} = \{s^m w^m n^m o^m \mid m \geq 1\}$ spezifiziert die Menge aller Quadrate, deren Seitenlänge ein Vielfaches der Einheitslänge ist:



Die Sprache L_{square} ist bekanntlich nicht kontextfrei, so daß für die Bildsprache aller Quadrate vorläufig unklar bleibt, ob sie kontextfrei ist oder nicht.

Die Beispiele belegen bereits einige Phänomene von Bildern und Bildsprachen, die durch Kettencodes hergestellt sind. Die Bilder setzen sich aus waagerechten und senkrechten Linien zusammen, deren Länge ein Vielfaches der Einheitslänge ist. Sie können in einem Zug gezeichnet werden und durch Berührungen und Überschneidungen mehrere Gebiete einschließen, deren Zahl für die Bilder einer Sprache unbeschränkt groß werden kann. Außerdem muß beachtet werden, daß verschiedene Zeichenketten dasselbe Bild darstellen können, daß die Linien eines Bildes mehrmals gezeichnet werden dürfen und daß insbesondere jede Linie rückwärts in den jeweiligen Gegenrichtungen durchlaufen werden kann. So repräsentieren die Zeichenketten *swnoonws*, *swnonosw*, *onwsswno*, *onwswson*, *noswwson*, *noswswno*, *wsonnosw*, *wsononws*, *swnoswnoonws*, *swnowoownosw*, *swonnowsonswwsno* u.v.a.m. alle dasselbe Bild, nämlich



2.2 Komposition von Bildern und Bildsprachen

Wie in anderen Bereichen der Informatik ist auch für Bilder und Bildsprachen von Interesse, ob und mit welchen Techniken sie sich aus Teilen modular aufbauen lassen. Einige noch sehr einfache, aber durchaus wirkungsvolle Operationen sind im folgenden diskutiert. Zum Beispiel ist die Vereinigung zweier Bildsprachen wieder eine Bildsprache. Interessanter jedoch ist das Hintereinanderhängen gezeichneter Bilder. Dazu verschiebt man das zweite Bild, so daß sein Anfangspunkt auf dem Endpunkt des ersten Bildes liegt. Danach entspricht die Vereinigung der Bilder dem Hintereinanderzeichnen. Darüber hinaus läßt sich jedes gezeichnete Bild beliebig oft mit sich selber auf diese Weise verketten. Beide Operationen lassen sich mühelos auf Mengen gezeichneter Bilder fortsetzen. Die Überlegungen zur Verkettung lassen sich folgendermaßen präzisieren:

2.3 Definition

1. Seien $x_0, y_0 \in \mathbb{Z}$. Dann lassen sich Punkte, Einheitslinien, Bilder und Bildsprachen um x_0 Einheiten in der x-Achse und y_0 Einheiten in der y-Achse verschieben. Im einzelnen ist die *Verschiebung* s_{x_0,y_0} definiert durch:

   (i) $s_{x_0,y_0}((x,y)) = (x + x_0, y + y_0)$ für alle $(x,y) \in \mathbb{Z}^2$,

   (ii) $s_{x_0,y_0}(l) = \{s_{x_0,y_0}((x,y)), s_{x_0,y_0}((x',y'))\}$ für alle $l = \{(x,y),(x',y')\} \in UL$,

   (iii) $s_{x_0,y_0}(p) = \{s_{x_0,y_0}(l) \mid l \in p\}$ für alle Bilder p,

   (iv) $s_{x_0,y_0}(B) = \{s_{x_0,y_0}(p) \mid p \in B\}$ für alle Bildmengen B.

2. Seien $d = (p,b,e)$ und $d' = (p',b',e')$ zwei gezeichnete Bilder.

   (i) Wähle $x_0, y_0 \in \mathbb{Z}$ so, daß $s_{x_0,y_0}(b') = e$. Dann ist die *Verkettung* von d und d' gegeben durch:

   $$d \bullet d' = (p \cup s_{x_0,y_0}(p'), b, s_{x_0,y_0}(e'))$$

   (ii) Die *i-te Verkettung* von d mit sich selbst ist rekursiv gebildet durch:

   (a) $d^0 = (\emptyset, (0,0), (0,0))$,

   (b) $d^{i+1} = d^i \bullet d$ für $i \in \mathbb{N}$.

3. Seien D und D' zwei Mengen gezeichneter Bilder.

   (i) Dann ist $D \bullet D' = \{d \bullet d' \mid d \in D, d' \in D'\}$ die *Verkettung* von D und D'.

   (ii) Die *i-te Verkettung* von D mit sich selbst ist rekursiv gebildet durch:

   (a) $D^0 = \{(\emptyset, (0,0), (0,0))\}$,

   (b) $D^{i+1} = D^i \bullet D$ für $i \in \mathbb{N}$.

   (iii) Ferner wird $D^* = \bigcup_{i \in \mathbb{N}} D^i$ der *Kleene-Stern* von D genannt. $\diamond$

Mit Hilfe der Vereinigung, der Verkettung und dem Kleene-Stern lassen sich nun aus einzelnen gezeichneten Bildern Mengen von gezeichneten Bildern aufbauen, die regulär genannt werden.

2.4 Definition
Die Menge der regulären Mengen von gezeichneten Bildern $\mathcal{R}$ ist rekursiv durch folgenden Prozeß aufgebaut:

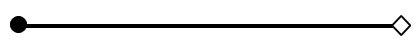
(i) $\emptyset \in \mathcal{R}$.

(i) Ist $d = (p, (0,0), e))$ ein gezeichnetes Bild, so gilt $\{d\} \in \mathcal{R}$.

(ii) Für alle $D, D' \in \mathcal{R}$ sind auch $D \cup D', D \bullet D', D^* \in \mathcal{R}$. $\diamond$

2.3 Beispiel

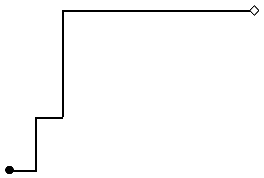
Nach Definition induzieren die beiden gezeichneten Bilder $\hat{o} = (\{(0,0),(1,0)\},(0,0),(1,0))$ und $\hat{n} = (\{(0,0),(0,1)\},(0,0),(0,1))$, die aus den Einheitslinien vom Ursprung des Koordinatensystems zum östlichen bzw. nördlichen Nachbarn führen, reguläre einelementige Mengen von gezeichneten Bildern, die mit $E(o)$ bzw. $E(n)$ bezeichnet werden. Dann haben $\hat{o}^7$ und $\hat{n}^4$ folgende Form, wobei die Einheitslänge 0,5 cm beträgt:

$\hat{o}^7 =$  $\hat{n}^4 =$

Und entsprechend enthalten die Kleene-Sterne $E(o)^*$ und $E(n)^*$ gerade alle Linien in östlicher bzw. nördlicher Richtung. Die Verkettung $E(n)^* \bullet E(o)^*$ enthält dann alle Stufen mit beliebiger Höhe und Tiefe, z.B.

$\hat{n}^4 \bullet \hat{o}^7 =$ 

Schließlich besteht der Kleene-Stern $(E(n)^* \bullet E(o)^*)^*$ aus allen Treppen mit beliebigen Stufen, z.B.

$(\hat{n}^0 \bullet \hat{o}^1) \bullet (\hat{n}^2 \bullet \hat{o}^1) \bullet (\hat{n}^4 \bullet \hat{o}^7) =$ 

Bis jetzt spielt sich die Behandlung von regulären Mengen von gezeichneten Bildern auf der Seite der Bilder ab. Eine wichtige Frage ist, ob sie auch durch Kettencodes beschreibbar und aufbaubar sind. Es stellt sich heraus, daß die Operationen, die Regularität auf der Bildseite etablieren, direkt mit entsprechenden Operationen auf der Seite der Bildbeschreibungen und damit zu regulären Mengen von Bildbeschreibungen korrespondieren. Damit steht eine Vielzahl von Beschreibungsmitteln (endliche Automaten, reguläre Grammatiken, reguläre Ausdrücke und andere) zur Verfügung, um reguläre Mengen von gezeichneten Bildern zu spezifizieren. Dieses Ergebnis ist in folgendem Theorem festgehalten.

2.5 Theorem
Eine Menge B von Bildern ist genau dann regulär, wenn es eine reguläre Menge D von gezeichneten Bildern gibt, so daß $B = \{p \mid (p,b,e) \in D$ für geeignete $b, e \in \mathbb{Z}^2\}$.

Beweis. Zunächst einmal ist es nützlich, sich zu vergewissern, daß die Verkettung von gezeichneten Bildern assoziativ ist. Für gezeichnete Bilder $d = (p,b,e)$, $d' = (p',b',e')$ und $d'' = (p'',b'',e'')$ gilt nämlich

$$
\begin{aligned}
(d \bullet d') \bullet d'' &= (p \cup s_e(p'), b, s_e(e')) \bullet d'' \\
&= (p \cup s_e(p') \cup s_{s_e(e')}(p''), b, s_{s_e(e')}(e'')) \\
&= (p \cup s_e(p') \cup s_e(s_{e'}(p'')), b, s_e(s_{e'}(e''))) \\
&= (p \cup s_e(p' \cup s_{e'}(p'')), b, s_e(s_{e'}(e''))) \\
&= (p,d,e) \bullet (p' \cup s_{e'}(p''), b, s_{e'}(e'')) \\
&= (p,d,e) \bullet (d' \bullet d'').
\end{aligned}
$$

Außerdem gilt für alle Wörter $u \in \{o,s,w,n\}^*$ und alle $x \in \{o,s,w,n\}$

$$
\begin{aligned}
\mathit{drawing}(ux) &= (p(u) \cup s_{e(u)}(p(x)), b(u), s_{e(u)}(e(x))) \\
&= (p(u), b(u), e(u)) \bullet (p(x), b(x), e(x)) \\
&= \mathit{drawing}(u) \bullet \mathit{drawing}(x).
\end{aligned}
$$

Mit Hilfe dieser beiden Beziehungen kann man nun zeigen, daß die Verkettung zweier Bildbeschreibungen der Verkettung ihrer gezeichneten Bilder entspricht: Für alle Bildbeschreibungen u, v gilt $\mathit{drawing}(uv) = \mathit{drawing}(u) \bullet \mathit{drawing}(v)$. Der Beweis erfolgt per Induktion über $\mathit{length}(v)$.

Für $v = \lambda$ ergibt sich direkt $\mathit{drawing}(uv) = \mathit{drawing}(u) = \mathit{drawing}(u) \bullet \mathit{drawing}(v)$. Sei nun $v = wx$ und nimm an, die Behauptung gelte für u und w. Dann ergibt sich

$$
\begin{aligned}
\mathit{drawing}(uv) &= \mathit{drawing}(u(wx)) \\
&= \mathit{drawing}((uw)x) \\
&= \mathit{drawing}(uw) \bullet \mathit{drawing}(x) \\
&= (\mathit{drawing}(u) \bullet \mathit{drawing}(w)) \bullet \mathit{drawing}(x) \\
&= \mathit{drawing}(u) \bullet (\mathit{drawing}(w) \bullet \mathit{drawing}(x)) \\
&= \mathit{drawing}(u) \bullet \mathit{drawing}(wx) \\
&= \mathit{drawing}(u) \bullet \mathit{drawing}(v),
\end{aligned}
$$

wobei die dritte, fünfte und sechste Gleichheit oben gezeigt wurden und die vierte die Induktionsvoraussetzung ist.

Jetzt kann die Behauptung des Theorems selbst gezeigt werden. Genauer gesagt, wird eine geringfügig stärkere Behauptung gezeigt:

> Eine Menge D von gezeichneten Bildern ist genau dann regulär, wenn es einen regulären Ausdruck A gibt, so daß $D = \{\mathit{drawing}(u) \mid u \in L(A)\}$.

(⇒) Per Induktion über den Aufbau von D. Falls $D = \emptyset$ oder $D = \{\mathit{drawing}(u)\}$ für eine einzelne Bildbeschreibung u, so wähle ein A mit $L(A) = \emptyset$ bzw. $L(A) = \{u\}$ (was bekanntermaßen immer möglich ist).

Betrachte nun reguläre Mengen D_1, D_2 gezeichneter Bilder und nimm als Induktionsvoraussetzung an, daß es reguläre Ausdrücke A_1, A_2 mit $D_i = \{\mathit{drawing}(u) \mid u \in L(A_i)\}$ gibt. Gemäß der Definition regulärer Mengen gezeichneter Bilder sind drei Fälle zu unterscheiden:

1. Für $D = D_1 \cup D_2$ wähle $A = A_1 + A_2$. Dann folgt nach Definition von $L(A)$ wie gewünscht

$$\begin{aligned} D &= \{\mathit{drawing}(u) \mid u \in L(A_1)\} \cup \{\mathit{drawing}(u) \mid u \in L(A_2)\} \\ &= \{\mathit{drawing}(u) \mid u \in L(A_1) \cup L(A_2)\} \\ &= \{\mathit{drawing}(u) \mid u \in L(A)\}. \end{aligned}$$

2. Für $D = D_1 \bullet D_2$ wähle $A = A_1 \circ A_2$. Mit der oben gezeigten Gleichung ergibt sich dann

$$\begin{aligned} D &= \{d_1 \bullet d_2 \mid d_1 \in D_1, d_2 \in D_2\} \\ &= \{\mathit{drawing}(u_1) \bullet \mathit{drawing}(u_2) \mid u_1 \in L(A_1), u_2 \in L(A_2)\} \\ &= \{\mathit{drawing}(u_1 u_2) \mid u_1 \in L(A_1), u_2 \in L(A_2)\} \\ &= \{\mathit{drawing}(u) \mid u \in L(A)\}. \end{aligned}$$

3. Für $D = D_1^*$ wähle $A = A_1^*$. Der Einfachheit halber benutzen wir die leicht per Induktion nachzuweisende Gleichung $D = \{d_1 \bullet \cdots \bullet d_n \mid d_1, \ldots, d_n \in D_1 \text{ für ein } n \in \mathbb{N}\}$ (wobei in Anbetracht der oben gezeigten Assoziativität die Klammern wegegelassen wurden und $d_1 \bullet \cdots \bullet d_n = (\emptyset, (0,0), (0,0))$ für $n = 0$), anstelle der eigentlichen Definition des Kleene-Sterns. Damit ergibt sich

$$\begin{aligned} D &= \{\mathit{drawing}(u_1) \bullet \cdots \bullet \mathit{drawing}(u_n) \mid u_1, \ldots, u_n \in L(A_1), n \in \mathbb{N}\} \\ &= \{\mathit{drawing}(u_1 \cdots u_n) \mid u_1, \ldots, u_n \in L(A_1), n \in \mathbb{N}\} \\ &= \{\mathit{drawing}(u) \mid u \in L(A)\}, \end{aligned}$$

was den Beweis dieser Richtung abschließt.

($\Leftarrow$) Hier geht die Induktion über den Aufbau des regulären Ausdrucks A. Für $A = \mathit{EMPTY}$ gilt $D = \emptyset$, was nach Definition eine reguläre Menge gezeichneter Bilder ist. Für $A = \mathit{LAMBDA}$ oder $A \in \{o, s, w, n\}$ ist $L(A)$ eine einelementige Menge $\{u\}$, so daß nach Definition $D = \{\mathit{drawing}(u)\}$ ebenfalls regulär ist. Schließlich sind im Induktionsschluß die Fälle $A = A_1 + A_2$, $A = A_1 \circ A_2$ und $A = A_1^*$ zu betrachten, wobei als Induktionsvoraussetzung angenommen wird, daß $D_i = \{\mathit{drawing}(u) \mid u \in L(A_i)\}$ für $i \in \{1, 2\}$ eine reguläre Menge gezeichneter Bilder ist. Wählt man analog zu der obigen Vorgehensweise $D = D_1 \cup D_2$, $D = D_1 \bullet D_2$ bzw. $D = D_1^*$, so erhält man aufgrund der schon oben bewiesenen Gleichungen wie benötigt $D = \{\mathit{drawing}(u) \mid u \in L(A)\}$, was den Beweis abschließt. ∎

2.4 Mitgliedschaftsproblem

In allen Situationen der Informatik, in denen endliche syntaktische Gebilde eingeführt werden, um die eigentlich interessierenden semantischen Objekte zu spezifizieren, besteht eine grundsätzliche Frage darin, welche Informationen ein syntaktisches Gebilde über das beschriebene Objekt liefert. Die Frage wird umso drängender, je weniger intuitiv die Beziehung

zwischen Syntax und Semantik ist. Bei Kettencode-Bildsprachen ist das Problem besonders virulent, denn für eine gegebene Grammatik beispielsweise mag es noch relativ klar sein, welche Zeichenketten erzeugt werden, aber welche Bilder dadurch entstehen, wird i.a. nicht bekannt sein. Deshalb ist es in diesem Zusammenhang interessant, Fragen über die erzeugten Bilder an die erzeugenden syntaktischen Beschreibungen zu stellen. Typische Fragen dieser Art sind:

(a) Wird überhaupt ein Bild erzeugt?

(b) Werden unendlich viele Bilder erzeugt?

(c) Wird ein bestimmtes Bild erzeugt?

(d) Sind unter den erzeugten Bildern solche, deren Linien Gebiete einschließen?

(e) Enthalten alle erzeugten Bilder Richtungsänderungen?

(f) Dehnen sich die erzeugten Bilder beliebig weit in östliche Richtung aus?

Als ein erstes Beispiel, wie man mit solchen Fragen umgehen kann, wird gezeigt, daß die dritte Frage für kontextfreie Kettencode-Bildsprachen algorithmisch beantwortet werden kann, also entscheidbar ist. Dabei werden zwei bekannte Ergebnisse aus der Theorie formaler Sprachen angewendet, nämlich daß der Durchschnitt einer kontextfreien mit einer regulären Sprache wieder kontextfrei, und daß das Leerheitsproblem lösbar ist. Um dieses Faktum ausnutzen zu können, wird als Hilfssatz nachgewiesen, daß alle Bildbeschreibungen eines gezeichneten Bildes eine reguläre Sprache formen.

2.6 Theorem
Für alle kontextfreien Grammatiken G (mit dem terminalen Alphabet $\{o, s, w, n\}$) und für alle gezeichneten Bilder $d = (p, (0,0), e)$ ist die Frage entscheidbar, ob ein $u \in L(G)$ existiert mit $\mathit{drawing}(u) = d$.

Beweis. Nach dem folgenden Lemma ist die Menge $\mathit{descript}(d)$ aller Bildbeschreibungen von d regulär, wobei im Lemma explizit aus d ein endlicher Automat $A(d)$ konstruiert wird, der $\mathit{descript}(d)$ erkennt. Nach einem bekannten Satz der Theorie formaler Sprachen ist dann aus G und A eine kontextfreie Grammatik $G(A(d))$ konstruierbar mit

$$L(G(A(d))) = L(G) \cap L(A(d)) = L(G) \cap \mathit{descript}(d).$$

Diese Grammatik kann dann als Eingabe für eine Lösung des Leerheitsproblems kontextfreier Sprachen verwendet werden. Bei positiver Antwort gibt es eine Bildbeschreibung $u \in L(G(A(d)))$ und damit $u \in L(G)$ sowie $\mathit{drawing}(u) = d$. Im negativen Fall beschreibt keine Bildbeschreibung aus $L(G)$ das gezeichnete Bild d. Somit ist die Entscheidung des Theorems algorithmisch getroffen. ∎

2.7 Lemma
Sei $d = (p, (0,0), e)$ ein gezeichnetes Bild und

$$\mathit{descript}(d) = \{v \in \{o, s, w, n\}^* \mid \mathit{drawing}(v) = d\}$$

die Menge aller Bildbeschreibungen von d. Dann ist $\mathit{descript}(d)$ eine reguläre Sprache.

Beweis. Betrachte den folgenden endlichen Automaten $A(d) = (I, S, \delta, s_0, F)$ mit

- $I = \{o, s, w, n\}$ als Eingabealphabet,
- $S = (point(p) \times \mathcal{P}(p)) \cup \{\$\}$ als Zustandsmenge, wobei $\mathcal{P}(p)$ die Potenzmenge von p bezeichnet und \$ ein Extrasymbol ist,
- $\delta : I \times S \to S$ als Zustandsüberführung, die für alle $r \in \{o, s, w, n\}, (x, y) \in points(p)$, $Z \subseteq \mathcal{P}(p)$ und für \$ gegeben ist durch
- $\delta(r, ((x, y), Z)) = (r((x, y)), Z \cup \{\{(x, y), r((x, y))\}\})$, falls $\{(x, y), r((x, y))\} \in p$,
  $\delta(r, ((x, y), Z)) = \$$ sonst sowie
  $\delta(r, \$) = \$$,
- $s_0 = ((0, 0), \emptyset)$ als Anfangszustand, sowie
- $F = \{(e, p)\}$ als einelementige Menge von Endzuständen.

Es bleibt $L(A(d)) = descript(d)$ zu zeigen. Dabei wird folgende Beobachtung über die fortgesetzte Zustandsüberführung benutzt, die später bewiesen wird:

$$\delta^*(u, s_0) = \begin{cases} (e(u), p(u)) & \text{falls } p(u) \subseteq p \\ \$ & \text{sonst} \end{cases}$$

Sei nun $v \in L(A(d))$. Das bedeutet $\delta^*(v, s_0) \in F$, also $\delta^*(v, s_0) = (e, p)$. Nach obiger Beobachtung ist das genau dann der Fall, wenn $e = e(v)$ und $p = p(v)$, was aber gerade bedeutet, daß

$$drawing(v) = (p(v), b(v), e(v)) = (p, (0, 0), e) = d$$

und damit $v \in descript(v)$.

Ist umgekehrt $v \in descript(v)$, dann gilt $drawing(v) = (p(v), b(v), e(v)) = d = (p, (0, 0), e)$ und damit $p(v) = p$ und $e(v) = e$. Insbesondere ist $p(v) \subseteq p$ und somit $\delta^*(v, s_0) = (e(v), p(v)) = (e, p)$. Nach Definition ist das der Endzustand, und deshalb gilt: $v \in L(A(d))$.

Es bleibt die obige Beobachtung zu beweisen, was durch vollständige Induktion über die Länge von u gemacht werden kann.

Induktionsanfang: $\delta^*(\lambda, s_0) = s_0 = ((0, 0), \emptyset) = (e(\lambda), p(\lambda))$, was wegen $p(\lambda) = \emptyset \subseteq p$ immer gilt.

Induktionsvoraussetzung: Die Behauptung gelte für u.

Induktionsschluß : Sei $r \in \{o, s, w, n\}$. Es gilt: $\delta^*(ur, s_0) = \delta(r, \delta^*(u, s_0))$. Nach Induktionsvoraussetzung ist $\delta^*(u, s_0) = (e(u), p(u))$ falls $p(u) \subseteq p$ und $\delta^*(u, s_0) = \$$ andernfalls (d.h. $p(u) \not\subseteq p$). Im ersten Fall erhält man: $\delta(r, \delta^*(u, s_0)) = \delta(r, (e(u), p(u))) = (r(e(u)), p(u) \cup \{\{e(u), r(e(u))\}\}) = (e(ur), p(ur))$ falls $\{e(u), r(e(u))\} \in p$, d.h. $\delta^*(ur, s_0) = (e(ur), p(ur))$ falls $p(ur) \subseteq p$. Ist allerdings $\{e(u), r(e(u))\} \notin p$ und damit auch $p(ur) \not\subseteq p$, dann gilt nach Definition von δ : $\delta^*(ur, s_0) = \delta(r, (e(u), p(u))) = \$$. Ist schließlich $p(u) \not\subseteq p$ und damit erst recht $p(ur) \not\subseteq p$, dann folgt: $\delta^*(ur, s_0) = \delta(r, \delta^*(u, s_0)) = \delta(r, \$) = \$$. Alles zusammen verifiziert die Behauptung für ur. ∎

2.5 Berechnung der Endpunkte

Die generelle Frage ist, ob sich anhand von Bildbeschreibungen Aussagen über die zugehörigen Bilder machen lassen (ohne diese unbedingt zeichnen zu müssen). Welche Möglichkeiten dabei bestehen, soll am Beispiel des Endpunktes bzw. der Ermittlung seiner Koordinaten demonstriert werden.

In der ersten Variante muß man beachten, daß für jedes Zeichen der Bildbeschreibung eine der beiden Koordinaten des gerade aktuellen Punktes um Eins erhöht oder erniedrigt wird. Man erhält also die Koordinaten des Endpunkts, indem man die Vorkommen aller vier Himmelsrichtungen zählt und die entsprechenden Differenzen bildet.

2.8 Beobachtung
Sei $u \in \{o, s, w, n\}^*$. Dann gilt: $e(u) = (\#_o(u) - \#_w(u), \#_n(u) - \#_s(u))$. ◇

Die zweite Vorgehensweise ist komplizierter, dafür aber abgewandelt für andere Probleme nutzbar. Sie geht von der Beobachtung aus, daß die Koordinaten des Endpunkts bei bestimmten Bildbeschreibungen unmittelbar abgelesen werden können:

$$\begin{aligned} e(o^i n^j) &= (i, j) \\ e(o^i s^j) &= (i, -j) \\ e(w^i n^j) &= (-i, j) \\ e(w^i s^j) &= (-i, -j). \end{aligned}$$

Wenn es nun gelänge, eine beliebige Bildbeschreibung so in eine dieser speziellen Form umzuwandeln, daß sich dabei der Endpunkt nicht ändert, wäre das Problem gelöst. Solche Umwandlungen lassen sich tatsächlich durch geeignete Chomsky-Regeln bewerkstelligen, was hier wegen der größeren Transparenz in zwei Schritten erfolgt.

Als Erstes werden Bildbeschreibungen so sortiert, daß alle horizontalen Richtungen vor allen vertikalen stehen. Dafür können die Regeln

$$\begin{aligned} so &::= os \\ sw &::= ws \\ no &::= on \\ nw &::= wn \end{aligned}$$

verwendet werden, die die Regelmenge P_{sort} bilden. Wendet man diese Regeln solange wie möglich an, wird jede Bildbeschreibung in eine umgewandelt, in der keine horizontale Richtung mehr hinter einer vertikalen vorkommt. Außerdem bleibt beim Regelanwenden der Endpunkt erhalten. Um die Sortiereigenschaft zu zeigen, wird für jede Bildbeschreibung $u = r_1 \cdots r_m$ ($r_i \in \{o, s, w, n\}, i = 1, \ldots, m$) eine Bewertung $val(u)$ eingeführt, die angibt, wieviele Richtungen in u wie oft falsch sortiert sind:

$val(u) = \sum_{i=1}^{m} val_i$ mit $val_i = \#_s(r_1 \cdots r_{i-1}) + \#_n(r_1 \cdots r_{i-1})$ für $i > 0$ und $r_i \in \{o, w\}$ sowie $val_i = 0$ sonst.

2.9 Beobachtung
Sei $u \underset{P_{sort}}{\rightarrow} u'$. Dann gilt:

1. $e(u) = e(u')$.
2. $val(u) > val(u')$. ⋄

2.10 Beobachtung
Folgende Eigenschaften von $\overline{u} \in \{o, s, w, n\}^*$ sind äquivalent:

(i) $val(\overline{u}) = 0$.

(ii) $\overline{u}$ ist P_{sort}-reduziert.

(iii) Es existieren $v_1 \in \{o, w\}^*$ und $v_2 \in \{s, n\}^*$ mit $\overline{u} = v_1 v_2$.

Dabei wird eine Bildbeschreibung P_{sort}-reduziert genannt, wenn keine der Regeln mehr anwendbar ist. ⋄

Aus Beobachtung 2.9 folgt, daß jede Ableitung, die bei u beginnt, nach spätestens $val(u)$ Schritten in einer P_{sort}-reduzierten Bildbeschreibung endet, die denselben Endpunkt wie u hat. Nach Beobachtung 2.10 zerfällt das Ergebnis in zwei Teile, wobei der erste nur horizontale Richtungen enthält, der zweite nur vertikale.

Als Zweites werden nun Bildbeschreibungen um Zeichenschritte bereinigt, bei denen dieselbe Linie zweimal direkt hintereinander gezeichnet werden. Dafür kann die Regelmenge P_{erase} mit den Regeln

$$\begin{array}{lcl} ow & ::= & \lambda \\ wo & ::= & \lambda \\ sn & ::= & \lambda \\ ns & ::= & \lambda \end{array}$$

herangezogen werden. Die Regelanwendungen bewahren den Endpunkt und verkürzen offensichtlich. Außerdem sind die Regeln auf Bildbeschreibungen, die nur horizontale oder nur vertikale Richtungen enthalten, gerade dann nicht mehr anwendbar, wenn nur noch eine Richtung vorkommt.

2.11 Beobachtung
Sei $v \underset{P_{erase}}{\rightarrow} v'$. Dann gilt:

1. $e(v) = e(v')$.
2. $length(v) = length(v') + 2$. ⋄

2.12 Beobachtung

1. $\overline{v} \in \{o, w\}^*$ ist genau dann P_{erase}-reduziert, wenn ein $i \in \mathbb{N}$ existiert mit $\overline{v} = o^i$ oder $\overline{v} = w^i$.
2. $\hat{v} \in \{s, n\}^*$ ist genau dann P_{erase}-reduziert, wenn ein $j \in \mathbb{N}$ existiert mit $\hat{v} = s^j$ oder $\hat{v} = n^j$. ⋄

Nach Beobachtung 2.11 endet jeder Ableitung, die bei einer Bildbeschreibung v beginnt, nach k Schritten, wobei k höchstens die halbe Länge von v ist. Das Ergebnis enthält nur eine Richtung, wenn v horizontal und vertikal nicht mischt. Außerdem bleiben die Endpunkte erhalten. Zusammenfassend wird also durch beliebiges Regelanwenden, jeweils solange es geht, eine Ableitung der folgenden Form erreicht:

$$u \underset{P_{sort}}{\overset{*}{\rightarrow}} v_1 v_2 \underset{P_{erase}}{\overset{*}{\rightarrow}} r^i \overline{r}^j \text{ mit } v_1 \in \{o, w\}^*, v_2 \in \{s, n\}^*, r \in \{o, w\}, \overline{r} \in \{s, n\} \text{ und } i, j \in \mathbb{N}.$$

Wegen $e(u) = e(r^i \overline{r}^j)$ ergeben sich aus den Exponenten i und j die Koordinaten von $e(u)$ wie anfangs bemerkt.

Anhang

Es folgen die Beweise der letzten fünf Beobachtungen.

Beweis von Beobachtung 2.8 durch Induktion über den Aufbau von u:

(IA) $e(\lambda) = (0, 0) = (0 - 0, 0 - 0) = (\#_o(\lambda) - \#_w(\lambda), \#_n(\lambda) - \#_s(\lambda))$.

(IV) Die Behauptung gelte für u.

(IS) Betrachte ur für $r \in \{o, s, w, n\}$.

Wähle zuerst $r = o$. Dann gilt nach Definition des Endpunkts, nach Induktionsvoraussetzung, nach Definition des östlichen Nachbarn sowie der Richtungszählung:

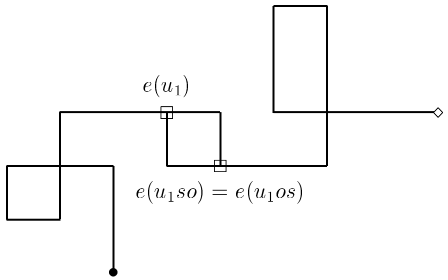
$$\begin{aligned} e(uo) = o(e(u)) &= o((\#_o(u) - \#_w(u), \#_n(u) - \#_s(u))) \\ &= ((\#_o(u) - \#_w(u) + 1, \#_n(u) - \#_s(u)) \\ &= ((\#_o(uo) - \#_w(uo), \#_n(uo) - \#_s(uo)). \end{aligned}$$

Analog kann man für s, w und n argumentieren, so daß insgesamt die Behauptung für ur gezeigt ist.

Beweis von Beobachtung 2.9:

1. Betrachte den Fall der Anwendung der ersten Regel.

Dann hat u die Form $u_1 so\, u_2$ und $u' = u_1 os\, u_2$. Nach Zeichnen von u_1 ist in beiden Fällen der Punkt $e(u_1) = (x, y)$ erreicht. Durch Lesen der nächsten beiden Zeichen wird daraus im Falle von u $(x, y - 1)$ und $(x + 1, y - 1)$ und im Falle u' $(x + 1, y)$ und $(x + 1, y - 1)$. Von diesem Punkt aus wird wieder dasselbe Wort u_2 abgearbeitet und damit derselbe Endpunkt erreicht. Bei den anderen Regeln kann analog argumentiert werden.



$u_1 = nnwwsonnoo \qquad u_2 = oonnnwssooo$

2. Für eine horizontale Richtung wird eine Fehlsortierung durch Regelanwendung beseitigt. Daraus folgt die Behauptung, wenn man sich davon überzeugt, daß keine zusätzlichen Fehlsortierungen entstehen.

Beweis von Beobachtung 2.10:

(i) $\Rightarrow$ (ii) durch $\neg$ (ii) $\Rightarrow \neg$ (i): Wenn auf $\overline{u}$ eine Regel anwendbar ist, enthält $\overline{u}$ so, sw, no oder nw als Teilwort. Wenn diese horizontale Richtung an der Stelle i_0 steht, dann ist $val_{i_0} \geq 1$ und damit $val(\overline{u}) \geq 1$.

(ii) $\Rightarrow$ (iii): Nach Voraussetzung kommt in $\overline{u}$ weder so, noch sw, no oder nw als Teilwort vor. Also befinden sich alle horizontalen vor allen vertikalen Richtungen.

(iii) $\Rightarrow$ (i): Offensichtlich gilt $val(v_1 v_2) = 0$ nach Definition von val.

Beweise der Beoachtungen 2.11 und 2.12:

Punkt 1 von Beobachtung 2.11 ergibt sich analog zu Punkt 1 von Beobachtung 2.9. Punkt 2 ist klar, da zwei Zeichen gelöscht werden. Beobachtung 2.12 ist ebenfalls offensichtlich. Denn in einer Bildbeschreibung mit horizontalen (vertikalen) Richtungen kommen genau dann beide Richtungen vor, wenn ow oder wo (sn oder ns) ein Teilwort ist, wenn also noch eine Regel anwendbar ist.

Kapitel 3

Turtle-Bildsprachen auf Basis von L-Systemen

In diesem Kapitel wird eine andere, ebenfalls auf der Interpretation von Zeichenketten beruhende Methode zur Erzeugung von Bildern vorgestellt.

3.1 Konzept der Turtle-Bildsprachen

Die Turtle-Geometrie, die dem Zeichensystem der Programmiersprache LOGO nachempfunden ist, erlaubt eine ähnliche graphische Interpretation von Zeichenketten, wie sie bei Kettencode-Bildsprachen bereits verwendet wird. Das heißt, daß die Zeichenketten wieder zeichenweise von links nach rechts bearbeitet werden und jedes gelesene Zeichen zu einem Kommando an ein Zeichengerät führt. Das Zeichengerät funktioniert allerdings anders. Es befindet sich wieder jeweils an einem Punkt, zeigt aber zusätzlich noch in eine aktuelle Richtung. Bei der „Schildkröte“ liegt dieser Punkt irgendwo auf ihrer Mittelachse, und der Kopf gibt die Richtung an. Bei dem Zeichen F kriecht die Schildkröte um eine Streckeneinheit d vorwärts und zeichnet dabei eine Linie vom ursprünglichen Punkt zum neuen aktuellen Punkt am Ende der Linie. Bei dem Zeichen f kriecht sie genauso vorwärts, zeichnet aber nicht. Beim Zeichnen $+$ dreht sie sich um einen Einheitswinkel δ um den aktuellen Punkt nach links, bei $-$ entsprechend nach rechts. Alle anderen Zeichen werden vorläufig ignoriert. Notiert man die aktuelle Richtung als Winkel zwischen der Mittelachse der Schildkröte und der x-Achse, erhält man folgende Interpretation von Zeichenketten als Bilder:

3.1 Definition (Zeichnen in der Turtle-Geometrie)
Sei V ein Alphabet. Dann ist für $u \in V^*$ das *gezeichnete Bild*, das aus einer Menge $d(u)$ von Linien und einer Position $pos(u) = (x(u), y(u), \alpha(u))$ besteht, rekursiv gegeben durch:

(i) $d(\lambda) = \emptyset$, $x(\lambda) = 0 = y(\lambda)$ und $\alpha(\lambda) = 90°$;

(ii) $d(uF) = d(u) \cup \{\{(x(u), y(u)), (x(uF), y(uF))\}\}$,
$x(uF) = x(u) + d \cdot \cos\alpha(u)$, $y(uF) = y(u) + d \cdot \sin\alpha(u)$ und $\alpha(uF) = \alpha(u)$;

(iii) $d(uf) = d(u)$ und $pos(uf) = pos(uF)$;

(iv) $d(u\pm) = d(u)$, $x(u\pm) = x(u)$, $y(u\pm) = y(u)$, $\alpha(u\pm) = \alpha(u) \pm \delta$ (wobei $\pm \in \{+, -\}$);

(v) $d(uX) = d(u)$ und $pos(uX) = pos(u)$ sonst.

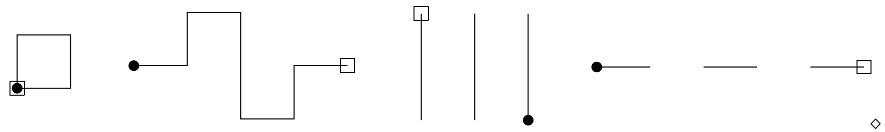
Dabei werden die Einheitslänge d und der Einheitswinkel δ vorab vereinbart. Wird nichts anderes gesagt, ist $d = 1$ und $\delta = 90°$. ◇

Zu beachten ist, daß der Anfangswinkel $\alpha(\lambda)$ 90° beträgt, die Schildkröte also zunächst nach oben (oder in Kettencode-Begriffen nach Norden) schaut.

3.2 Beispiele
Die Zeichenketten

$$F{-}F{-}F{-}F,$$
$$-F{+}F{-}F{-}FF{+}F{+}F{-}F,$$
$$FF{+}f{+}FF{-}f{-}FF \text{ und}$$
$$-FfFfF$$

ergeben folgende Muster, wobei Anfangs- und Endposition markiert sind:



◇

Wie im Kettencode-Ansatz kann mit Hilfe der Turtle-Interpretation jede Grammatik (oder jeder sonstige Mechanismus, der Zeichenkettensprachen beschreibt) als Spezifikation einer Bildsprache aufgefaßt werden, indem die Zeichenketten der erzeugten Sprache als Bilder gezeichnet werden. Bei der Turtle-Geometrie werden als Grammatiken hauptsächlich L-Systeme betrachtet, die von Lindenmayer zur Beschreibung biologischer Wachstumsprozesse eingeführt wurden. Da L-Systeme nicht bereits aus dem Grundstudium bekannt sind, werden Sie im folgenden kurz vorgestellt.

3.3 Definition (Lindenmayer-Systeme)

1. Ein *OL-System* $G = (V, \omega, P)$ besteht aus einem Alphabet V, einem *Axiom* $\omega \in V^*$ und einer endlichen *Regelmenge* $P \subseteq V \times V^*$. Für $(A, u) \in P$ wird auch $A \to u$ geschrieben.
2. Ist $a_1 \cdots a_n \in V^*$ und $a_i \to u_i \in P$ für $i = 1, \ldots, n$, dann ist $u_1 \cdots u_n$ aus $a_1 \cdots a_n$ mit P *direkt abgeleitet*, wofür auch $a_1 \cdots a_n \to u_1 \cdots u_n$ geschrieben wird.
3. Eine *Ableitung* $v_1 \to v_2 \to \cdots \to v_k$ besteht aus direkten Ableitungen $v_j \to v_{j+1}$ für $j = 1, \ldots, k-1$, wofür auch kurz $v_1 \overset{*}{\to} v_k$ geschrieben werden kann.
4. Die von G *erzeugte (Zeichenketten-)Sprache* enthält alle aus dem Axiom ableitbaren Zeichenketten:
$$L(G) = \{v \in V^* \mid \omega \overset{*}{\to} v\}.$$
5. Ein *deterministisches OL-System*, kurz: DOL-System, ist ein OL-System $G = (V, \omega, P)$, derart daß für jedes $a \in V$ genau ein $u \in V^*$ mit $a \to u \in P$ existiert. ◇

Das wichtigste Prinzip von L-Systemen besteht also darin, daß in jedem Ableitungsschritt jedes Zeichen der aktuellen Kette durch die rechte Seite einer passenden Regel ersetzt wird. Bei DOL-Systemen ist dieser Vorgang eindeutig bestimmt und kann immer weiter fortgesetzt werden. Beginnt man also mit $v_0 = \omega$, dann ist für jedes $n \geq 1$ durch die Iteration $v_{n-1} \rightarrow v_n$ ein Mitglied der erzeugten Sprache bestimmt, das man die *n-te Generation* nennen kann. Durch Angabe von n ist also im Falle von DOL-Systemen eindeutig ein Bild (nämlich $d(v_n)$) spezifiziert. Im folgenden werden vor allem DOL-Systeme betrachtet. Der Einfachheit halber werden reproduzierende Regeln der Form $a \rightarrow a$ nicht explizit angegeben (sofern für die Ersetzung von a nicht noch zusätzlich andere Regeln vorhanden sind).

3.4 Beispiele
Die in Abbildung 3.1 angegebenen Regelsysteme sind jeweils DOL-Systeme, die bis auf das letzte aus [PL90] stammen. Das letzte ist ein (um 90° gedrehtes) Beispiel aus [Edg90]. Man beachte, daß kleine Variationen in der rechten Regelseite sich bereits deutlich in den gezeichneten Bildern auswirken können. ◇

Die Turtle-Interpretation ist bisher vor allem in der Biologie angewendet worden, um insbesondere das Wachstum und den strukturellen Aufbau von Pflanzen und einfachen Zellgebilden zu beschreiben und sichtbar zu machen. Die Weiterentwicklung dieses Ansatzes ist deshalb vor allem biologisch motiviert, die dabei entstandenen Konzepte sind aber meist auch von allgemeinerem Interesse, wie das Modellieren von Verzweigungsstrukturen, das Füllen (und Färben) von Flächen und die Einbeziehung der dritten Dimension. Hier wird zunächst die Verzweigungstechnik erläutert, weil sie einen besonders engen Zusammenhang zur Informatik aufweist. Beim Verzweigen entstehen Bäume bzw. baumartige Strukturen, die sich bekanntlich mit der „depth-first"Strategie auf der Basis der Kellerspeicherung traversieren lassen. Die Schildkröte führt bei ihrer Arbeit einen Keller mit, der anfangs leer ist und folgendermaßen benutzt wird: Beim Lesen des Zeichens [wird die aktuelle Position auf dem Keller gespeichert; beim Lesen des Zeichens] wird der oberste Kellereintrag dort entfernt und zur neuen aktuellen Position der Schildkröte. Das bis dahin gezeichnete Bild bleibt unverändert. Im ersten Fall ändert sich auch die aktuelle Position nicht. Formal wird diese Überlegung durch den Zustandsbegriff erfaßt, der Definition 3.1 ergänzt.

3.5 Definition (Turtle-Zustand)
Sei V ein Alphabet (das zumindest die Zeichen $F, f, +, -, [,]$ enthält). Dann besteht für $u \in V^*$ der Zustand $state(u)$ der Schildkröte aus dem gezeichneten Bild $d(u)$ mit der Position $pos(u)$ und dem Keller $stack(u)$, die rekursiv definiert sind durch:

(i) $d(\lambda) = \emptyset$, $x(\lambda) = 0 = y(\lambda)$, $\alpha(\lambda) = 90°$ und $stack(\lambda) = \lambda$.

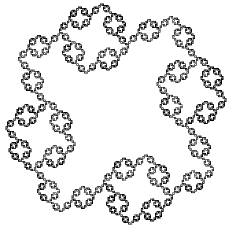
(ii) $d(uF) = d(u) \cup \{\{(x(u), y(u)), (x(uF), y(uF))\}\}$, $x(uF) = x(u) + d \cdot \cos\alpha(u)$, $y(uF) = y(u) + d \cdot \sin\alpha(u)$, $\alpha(uF) = \alpha(u)$ und $stack(uF) = stack(u)$.

(iii) $d(uf) = d(u)$, $pos(uf) = pos(uF)$ und $stack(uf) = stack(u)$.

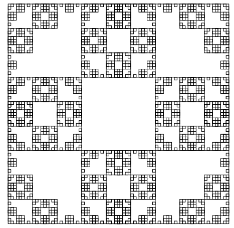
(iv) $d(u\pm) = d(u)$, $x(u\pm) = x(u)$, $y(u\pm) = y(u)$, $\alpha(u\pm) = \alpha(u) \pm \delta$ und $stack(u\pm) = stack(u)$.

(v) $d(u[) = d(u)$, $pos(u[) = pos(u)$ und $stack(u[) = push(stack(u), pos(u))$.

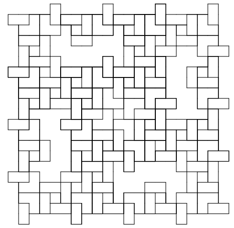
(vi) $d(u]) = d(u)$, $pos(u]) = top(stack(u))$ und $stack(u]) = pop(stack(u))$.



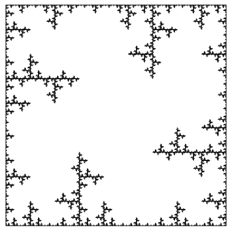
$n = 4,\ \delta = 90°$
$\omega = F{-}F{-}F{-}F$
$F \rightarrow FF{-}F{-}F{-}F{-}F{-}F{+}F$



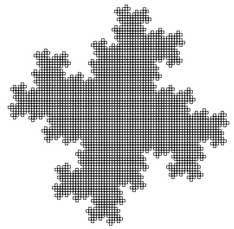
$n = 4,\ \delta = 90°$
$\omega = F{-}F{-}F{-}F$
$F \rightarrow FF{-}F{-}F{-}F{-}FF$



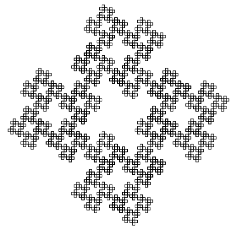
$n = 3,\ \delta = 90°$
$\omega = F{-}F{-}F{-}F$
$F \rightarrow FF{-}F{+}F{-}F{-}FF$



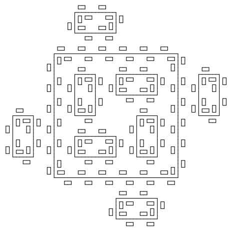
$n = 5,\ \delta = 90°$
$\omega = F{-}F{-}F{-}F$
$F \rightarrow FF{-}F{-}{-}F{-}F$



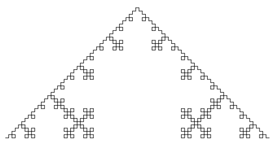
$n = 5,\ \delta = 90°$
$\omega = F{-}F{-}F{-}F$
$F \rightarrow F{-}FF{-}{-}F{-}F$



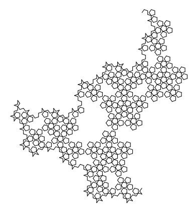
$n = 5,\ \delta = 90°$
$\omega = F{-}F{-}F{-}F$
$F \rightarrow F{-}F{+}F{-}F{-}F$



$n = 2,\ \delta = 90°$
$\omega = F{+}F{+}F{+}F$
$F \rightarrow F{+}f{-}FF{+}F{+}FF{+}Ff{+}FF$
$\quad {-}f{+}F{-}F{-}FF{-}FF{-}fFFF$
$f \rightarrow ffffff$



$n = 4,\ \delta = 90°$
$\omega = {-}F$
$F \rightarrow F{+}F{-}F{-}F{+}F$

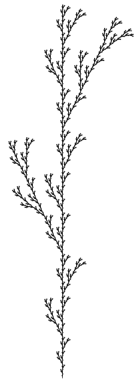


$n = 4,\ \delta = 36°$
$\omega = F$
$F \rightarrow {+}F{+}{+}F{-}{-}{-}{-}F$
$\quad {-}{-}F{+}{+}F{+}{+}F{-}$

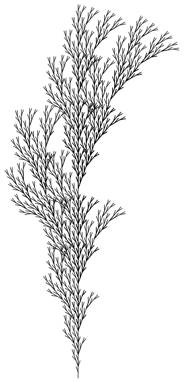
Abbildung 3.1: DOL-Systeme mit erzeugten Bildern

(vii) $d(uX) = d(u)$, $pos(uX) = pos(u)$ und $stack(uX) = stack(u)$ sonst. $\diamond$

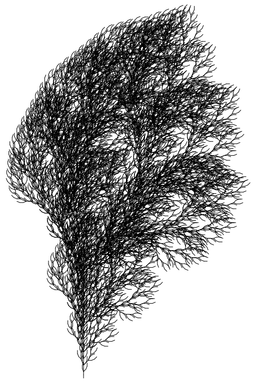
Dabei sind *push*, *pop*, und *top* die üblichen Operationen auf Kellern. Die in Abbildung 3.2 gezeigten Beispiele sind DOL-Systeme, die mit Verzweigungen arbeiten. Die erzielten Bilder ergeben bereits große Ähnlichkeiten mit bestimmten Pflanzenarten.



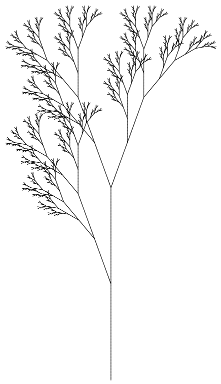
$n = 5,\ \delta = 25.7°$
$\omega = F$
$F \to F[+F]F[-F]F$



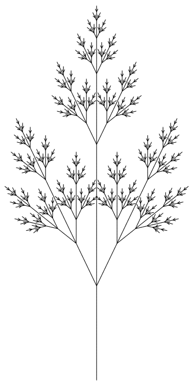
$n = 5,\ \delta = 20°$
$\omega = F$
$F \to F[+F]F[-F][F]$



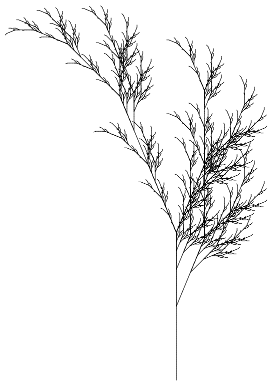
$n = 5,\ \delta = 20°$
$\omega = F$
$F \to FF-[-F+F+F]+[+F-F-F]$



$n = 7,\ \delta = 20°$
$\omega = X$
$X \to F[+X]F[-X]+X$
$F \to FF$



$n = 7,\ \delta = 25.7°$
$\omega = X$
$X \to F[+X][-X]FX$
$F \to FF$



$n = 6,\ \delta = 22.5°$
$\omega = X$
$X \to F-[[X]+X]+F[+FX]-X$
$F \to FF$

Abbildung 3.2: DOL-Systeme mit Verzweigungen zum Modellieren von Pflanzen

Die Turtle-Interpretation in Verbindung mit L-Systemen ist ausführlich in [PL90] dargestellt, in dem auch die Verallgemeinerung auf drei Dimensionen und die Einbeziehung von Fläche und Farbe und vieles andere mehr gefunden werden kann. Alle Konzepte sind eindrucksvoll durch biologische, insbesondere botanische Beispiele illustriert.

3.2 Von der Sturheit der Schildkröte

Eine der zentralen Fragen der Informatik betrifft die Beziehung zwischen Syntax und Semantik: Was verrät ein syntaktisches Gebilde (sei es ein Programm, eine Spezifikation, eine Grammatik, ein regelbasiertes System, ein Automat oder sonst irgendein formales endliches Objekt) über das dadurch Beschriebene (sei es eine berechenbare Funktion, eine erzeugte oder erkannte Sprache oder sonst irgendein oft unendliches Gebilde)? Die betrübliche Erkenntnis ist, daß solche Fragen meist nicht allgemein algorithmisch beantwortet werden können, also unentscheidbar sind. Entscheidbar werden solche Fragen erst bei speziellen Vorgaben, die dann oft so restriktiv sind, daß das eigentlich Interessierende nicht mehr beschrieben werden kann.

Bei der Turtle-Interpretation von L-Systemen verhält es sich leider auch so. Es stellt sich heraus, daß für sogenannte EOL-Systeme unentscheidbar ist, ob eines der gezeichneten Bilder verzweigt oder nicht. Dabei heißt Verzweigen, daß in einem Bildpunkt mindestens drei Bildlinien zusammentreffen. Genauer: Das gezeichnetete Bild $d(u)$ für $u \in V^*$ *verzweigt*, wenn es in $d(u)$ drei Linien l_1, l_2 und l_3 gibt, die paarweise genau einen Punkt p gemeinsam haben: $l_1 \cap l_2 = l_2 \cap l_3 = l_1 \cap l_3 = \{p\}$.

Um von einem vorliegenden Problem die Unentscheidbarkeit zu beweisen, benutzt man häufig das Mittel der *Reduktion*: Ein bekanntes unentscheidbares Problem wird in das gegebene übersetzt. Wäre das gegebene Problem entscheidbar, ergäbe sich ein Widerspruch zur Unentscheidbarkeit des bekannten Problems. Im Falle des Verzweigungsproblems für EOL-Systeme werden Postsche Korrespondenz-Probleme übersetzt, deren Lösbarkeit unentscheidbar ist.

Von einem OL-System unterscheidet sich ein EOL-System, bei dem das E für „extended“ (also „erweitert“) steht, durch die Unterscheidung terminaler von nichtterminalen Zeichen. Die erzeugte Sprache entsteht durch Herausfiltern der terminalen Wörter aus allen ableitbaren.

3.6 Definition (EOL-System)

1. Ein *EOL-System* $G = (V, T, \omega, P)$ besteht aus einem OL-System (V, ω, P) und einem terminalen Alphabet $T \subseteq V$.

2. Die von G erzeugte Sprache besteht aus allen aus dem Axiom ableitbaren terminalen Zeichenketten, wobei in G wie in (V, ω, P) abgeleitet wird:

$$L(G) = \{v \in T^* \mid \omega \xrightarrow{*} v\}. \qquad \blacksquare$$

Zur Erinnerung hier die Definition Postscher Korrespondenz-Probleme.

3.7 Definition (*Postsches Korrespondenz-Problem*)
Ein *Postsches Korrespondenz-Problem* über einem Alphabet A ist ein Paar $PCP = (U, V)$ gleichlanger endlicher Listen von Zeichenketten $U = (u_1, \dots, u_n)$ und $V = (v_1, \dots, v_n)$,

wobei $u_j, v_j \in A^*$ für $j = 1, \dots, n$. *PCP* ist *lösbar*, wenn es eine nichtleere Indexfolge $i_1 \cdots i_p \in \{1, \dots, n\}^*$ gibt mit $u_{i_1} \cdots u_{i_p} = v_{i_1} \cdots v_{i_p}$. ⋄

Es ist bekannt, daß die Lösbarkeit Postscher Korrespondenz-Probleme unentscheidbar ist, sobald A mindestens zwei Elemente enthält. Deswegen wird nun speziell für $A = \{0, 1\}$ zu jedem Postschen Korrespondenzproblem *PCP* ein EOL-System G_{PCP} konstruiert, derart daß *PCP* genau dann lösbar ist, wenn die Turtle-Bildsprache $\{d(w) \mid w \in L(G_{PCP})\}$ ein Bild enthält, das nicht verzweigt. Dabei codiert G_{PCP} in seinen Regeln korrespondierende Listeneinträge so, daß beim Ableiten die beiden Lösungsteile $u = u_{i_1} \cdots u_{i_p}$ und $v = v_{i_1} \cdots v_{i_p}$ als Treppen gezeichnet werden. Das Zeichen 0 entspricht einer senkrechten Linie von Einheitslänge und 1 entspricht einer waagerechten. Läuft man nun die u-Treppe hinauf und die v-Treppe hinunter und ist $u = v$, so kommt man beim Anfangspunkt ohne Verzweigen wieder an. Ist jedoch $i_1 \cdots i_p$ keine Lösung, weicht man irgendwann beim Hinunterlaufen von der u-Treppe ab, wobei eine Verzweigung entsteht, oder man kommt nicht bis zum Anfangspunkt zurück, oder man läuft darüber hinaus, oder die v-Treppe verläuft von Anfang an anders als die u-Treppe. Um diese drei Fälle auch noch in Verzweigungen zu verwandeln, wird an den Anfang und an das Ende sowohl der u-Treppe als auch der v-Treppe je eine diagonale Einheitslinie gehängt. Um das erreichen zu können, wird mit dem Einheitswinkel $\delta = 45°$ gearbeitet, der dann auch das Umkehren zwischen der u-Treppe und der v-Treppe ermöglicht.

3.8 Konstruktion
Sei $PCP = (U, V)$ mit $U = (u_1, \dots, u_n)$, $V = (v_1, \dots, v_n)$ und $u_i, v_j \in \{0, 1\}^*$ für alle $i, j \in \{1, \dots, n\}$. Dann ist

$$G_{PCP} = (\{S, T, F, +, -\}, \{F, +, -\}, S, P),$$

wobei P für $i \in \{1, \dots, n\}$ die folgenden Regeln enthält:

$$\begin{array}{lcl} S & \to & -F\,code(u_i)\,T\,edoc(v_i)\,F, \\ T & \to & F{+}{+}{+}{+}F \mid code(u_i)\,T\,edoc(v_i), \end{array}$$

sowie die identischen Ersetzungen $F \to F$, $+ \to +$ und $- \to -$. Die verwendeten Codierungen *code* und *edoc* sind rekursiv definiert durch

(i) $code(\lambda) = \lambda = edoc(\lambda)$,

(ii) $code(xu) = \overline{x}\,code(u)$ für $x \in \{0, 1\}$ und $u \in \{0, 1\}^*$ und

(iii) $edoc(xu) = edoc(u)\,\overline{x}$ für $x \in \{0, 1\}$ und $u \in \{0, 1\}^*$,

wobei $\overline{0} = +F-$ und $\overline{1} = -F+$. ⋄

Beachte, daß *code* und *edoc* jedes Zeichen einer Bitkette durch die korrespondierende Zeichenanweisung ersetzt, wobei *edoc* zusätzlich die Reihenfolge der Zeichen umkehrt. Das zusammen mit der terminalen rechten Seite für T, die eine Drehung um 180° und damit ein Umdrehen der Schildkröte bewirkt, erzeugt den Abstiegseffekt für die v-Treppe. Im einzelnen lassen sich für die konstruierten Gebilde folgende Eigenschaften zeigen:

3.9 Lemma

1. $code(uv) = code(u)code(v)$ für $u, v \in \{0, 1\}^*$.
2. $edoc(uv) = edoc(v)edoc(u)$ für $u, v \in \{0, 1\}^*$.
3. $L(G_{PCP})$ ist die Menge aller Wörter der Form
$$-F\,code(u_{i_1} \cdots u_{i_p})\,F{+}{+}{+}{+}F\,edoc(v_{i_1} \cdots v_{i_p})\,F$$
mit $p \geq 1$ und $i_j \in \{1, \dots, n\}$ für alle $j \in \{1, \dots, p\}$.
4. Betrachte $d(u\,code(w)\,F{+}{+}{+}{+}F\,edoc(w)\,v)$ und bezeichne mit $p(u)$ den Punkt $(x(u), y(u))$, auf dem die Schildkröte nach dem Zeichnen von u steht. Entsprechend bezeichne $p(z)$ den Punkt $(x(z), y(z))$, wobei $z = u\,code(w)\,F{+}{+}{+}{+}F\,edoc(w)$. Dann gilt $p(u) = p(z)$ und $\alpha(z) = \alpha(u) + 180°$, und auf dem Weg von $p(u)$ nach $p(z)$ liegt keine Verzweigung.
5. Betrachte $i_1 \cdots i_p$ mit $p \geq 1$ und $i_j \in \{1, \dots, n\}$ für $j = 1, \dots, p$. Für $u = u_{i_1} \cdots u_{i_p}$ und $v = v_{i_1} \cdots v_{i_p}$ verzweigt $d(-F\,code(u)\,F{+}{+}{+}{+}F\,edoc(v)\,F)$ genau dann nicht, wenn $u = v$.

Beweis. Die ersten beiden Punkte werden durch Induktion über den Aufbau von u mit Hilfe der Definition von *code* und *edoc* bewiesen.

1. I.A.: $code(\lambda v) = code(v) = \lambda\,code(v) = code(\lambda)code(v)$.
I.V.: Die Behauptung gelte für u (und beliebiges v).
I.S.: $code(xuv) = \overline{x}\,code(uv) = \overline{x}\,code(u)code(v) = code(xu)code(v)$.
2. I.A.: $edoc(\lambda v) = edoc(v) = edoc(v)\lambda = edoc(v)edoc(\lambda)$.
I.V.: Die Behauptung gelte für u (und beliebiges v).
I.S.: $edoc(xuv) = edoc(uv)\overline{x} = edoc(v)edoc(u)\overline{x} = edoc(v)edoc(xu)$.
3. Um dies zu beweisen, muß man sich die Ableitungsstruktur von G_{PCP} anschauen. Auf das Axiom S läßt sich eine der ersten n Regeln anwenden, auf das entstehende T kann eine der letzten n Regeln angewendet werden, was erneut ein T produziert, so daß dieser Vorgang $(p-1)$-mal fortgesetzt werden kann. Zum Schluß kann die terminierende Regel zum Zuge kommen; alle anderen Zeichen werden immer nur identisch ersetzt. Man erhält also:
$$\begin{array}{lcl} S & \to & -F\,code(u_{i_1})T\,edoc(v_{i_1})F \\ & \xrightarrow{*} & -F\,code(u_{i_1}) \cdots code(u_{i_p})T\,edoc(v_{i_p}) \cdots edoc(v_{i_1})F \\ & \to & -F\,code(u_{i_1}) \cdots code(u_{i_p})F{+}{+}{+}{+}F\,edoc(v_{i-p}) \cdots edoc(v_{i_1})F. \end{array}$$
Nach den ersten zwei Punkten läßt sich aus S also das Wort
$$-F\,code(u_{i_1} \cdots u_{i_p})F{+}{+}{+}{+}F\,edoc(v_{i_1} \cdots v_{i_p})F$$
ableiten.
4. Dieser Punkt kann für beliebige u und v durch vollständige Induktion über den Aufbau von w gezeigt werden.

I.A.: Aus $d(u\,code(\lambda)F{+}{+}{+}{+}F\,edoc(\lambda)v) = d(uF{+}{+}{+}{+}Fv)$ folgt nach Definition der Turtle-Geometrie $p(uF{+}{+}{+}{+}F) = p(u)$ und $\alpha(uF{+}{+}{+}{+}F) = \alpha(u) + 180°$.
I.V.: Die Behauptung gelte für w.
I.S.: Nach Definition von *code* und *edoc* gilt

$$d(u\,code(xw)F{+}{+}{+}{+}F\,edoc(xw)v) = d(u\overline{x}\,code(w)F{+}{+}{+}{+}F\,edoc(w)\overline{x}v)$$

und die Induktionsvoraussetzung liefert für $t = u\overline{x}code(w)F{+}{+}{+}{+}F\,edoc(w)$:

$$(*) \qquad x(u\overline{x}) = x(t),\ y(u\overline{x}) = y(t) \text{ und } \alpha(t) = \alpha(u\overline{x}) + 180°.$$

Betrachte nun $x = 0$. Nach Definition gezeichneter Bilder (wobei o.B.d.A. die Einheitslänge 1 angenommen wird) gilt:

$$\begin{aligned}
x(u\overline{0}) &= x(u{+}F{-}) = x(u) + \cos(\alpha(u) + 45°),\\
y(u\overline{0}) &= y(u{+}F{-}) = y(u) + \sin(\alpha(u) + 45°),\\
\alpha(u\overline{0}) &= \alpha(u{+}F{-}) = \alpha(u) + 45° - 45° = \alpha(u),\\
x(z) &= x(t\overline{0}) = x(t{+}F{-}) = x(t) + \cos(\alpha(t) + 45°),\\
y(z) &= y(t\overline{0}) = y(t{+}F{-}) = y(t) + \sin(\alpha(t) + 45°),\\
\alpha(z) &= \alpha(t\overline{0}) = \alpha(t{+}F{-}) = \alpha(t) + 45° - 45° = \alpha(t).
\end{aligned}$$

Mit $(*)$ und den Rechenregeln für sin und cos kann weiter umgeformt werden:

$$\begin{aligned}
x(z) &= x(t) + \cos(\alpha(t) + 45°) = x(u\overline{0}) + \cos(\alpha(u\overline{0}) + 225°)\\
&= x(u) + \cos(\alpha(u) + 45°) + \cos(\alpha(u) + 45° + 180°)\\
&= x(u) + \cos(\alpha(u) + 45°) - \cos(\alpha(u) + 45°) = x(u),\\
y(z) &= y(t) + \sin(\alpha(t) + 45°) = y(u\overline{0}) + \sin(\alpha(u\overline{0}) + 225°)\\
&= y(u) + \sin(\alpha(u) + 45°) + \sin(\alpha(u) + 45° + 180°)\\
&= y(u) + \sin(\alpha(u) + 45°) - \sin(\alpha(u) + 45°) = y(u),\\
\alpha(z) &= \alpha(t) = \alpha(u\overline{0}) + 180° = \alpha(u) + 180°.
\end{aligned}$$

Für $x = 1$ geht es analog. (Statt das alles formal auszurechnen, kann man sich auch die graphischen Effekte der beteiligten Kommandos verdeutlichen.)

5. Der fünfte Punkt ergibt sich nun relativ einfach aus dem dritten und vierten. Nach Punkt 3 ist $-F\,code(u)F{+}{+}{+}{+}F\,edoc(v)F \in L(G_{PCP})$. Wenn $u = v$ ist, dann gilt nach Punkt 4: $p(-F) = p(z)$ und $\alpha(z) = \alpha(-F) + 180°$ für $z = -F\,code(u){+}{+}{+}{+}edoc(u)$ und von $p(-F)$ nach $p(z)$ liegt keine Verzweigung. Nach Definition gezeichneter Bilder gilt außerdem für den Anfang und das Ende:

$$\begin{aligned}
x(-F) &= x(\lambda) + \cos(\alpha(\lambda) - 45°) = 0 + \cos(90° - 45°) = \cos 45°,\\
y(-F) &= y(\lambda) + \sin(\alpha(\lambda) - 45°) = 0 + \sin(90° - 45°) = \sin 45°,\\
\alpha(-F) &= \alpha(\lambda) - 45° = 45°,\\
x(zF) &= x(z) + \cos\alpha(z) = x(-F) + \cos(\alpha(-F) + 180°)\\
&= \cos 45° + \cos(45° + 180°) = \cos 45° - \cos 45° = 0,\\
y(zF) &= y(z) + \sin\alpha(z) = y(-F) + \sin(\alpha(-F) + 180°)\\
&= \sin 45° + \sin(45° + 180°) = \sin 45° - \sin 45° = 0.
\end{aligned}$$

Zwischen $p(\lambda) = (0,0)$ und $p(-F)$ und zwischen $p(z) = p(-F)$ und $p(zF) = (0,0)$ wird jeweils nur die diagonale Verbindungslinie gezogen, so daß weder in $(0,0)$ noch in $p(-F)$ eine Verzweigung stattfindet.

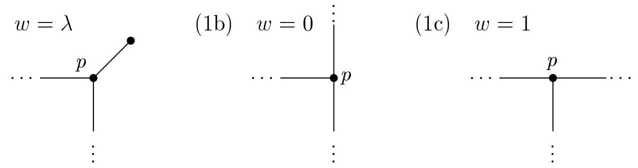
Ist andernfalls $u \neq v$, dann gibt es ein längstes Endstück w von u und v, d.h. $u = \overline{u}w$, $v = \overline{v}w$ und (1) $\overline{u}$ endet mit 1 und $\overline{v}$ mit 0 oder (2) $\overline{u}$ endet mit 0 und $\overline{v}$ mit 1 oder (3) $\overline{u} = \lambda$ und $\overline{v} \neq \lambda$ oder (4) $\overline{u} \neq \lambda$ und $\overline{v} = \lambda$.

Es gilt dann:

$$-F\,code(u){+}{+}{+}{+}edoc(v)F = -F\,code(\overline{u})code(w){+}{+}{+}{+}edoc(w)edoc(\overline{v})F.$$

Daraus läßt sich in allen vier Fällen eine Verzweigung in Punkt $p = p(-F\,code(\overline{u}))$ ermitteln. Für (1) ergeben sich folgende Situationen:

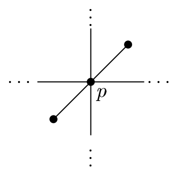
(1a) $w = \lambda$ (1b) $w = 0$ (1c) $w = 1$



Dabei kommt man von links mit $\overline{u}$ in p an, verläßt es mit w nach oben, nach rechts oder diagonal, kehrt zurück, um es mit $\overline{v}$ nach unten zu verlassen.

(2) analog, wobei man mit $\overline{u}$ von unten in p ankommt, mit w nach oben, nach rechts oder diagonal p verläßt und ein zweites Mal erreicht und dann mit $\overline{v}$ nach links verläßt.

(3) ergibt folgende Situation:



Dabei kommt man vom Nullpunkt auf der Diagonalen in p an, geht mit w nach oben, nach rechts oder diagonal weiter, kehrt nach p auf demselben Weg zurück und verläßt mit $\overline{v}\,p$ nach links oder nach unten.

(4) analog, wobei man von unten oder von links in p ankommt, mit w nach oben, nach rechts oder diagonal weitergeht und so zurückkehrt um dann diagonal nach unten p ein zweites Mal zu verlassen. ■

Aus den bisherigen Überlegungen ergibt sich nun recht mühelos ein Beweis für die Nichtentscheidbarkeit des Verzweigungsproblems für EOL-Systeme.

3.10 Theorem
Das Verzweigungsproblem für EOL-Systeme ist unentscheidbar, d.h. es gibt keinen Algorithmus, der für jedes EOL-System G entscheidet, ob alle gezeichneten Bilder $d(w)$ für $w \in L(G)$ verzweigen.

Beweis. Angenommen, es gäbe einen solchen Algorithmus. Dann könnte man insbesondere G_{PCP} für beliebiges PCP eingeben. Ist die Antwort negativ, dann gibt es ein $w \in L(G_{PCP})$, so daß $d(w)$ nicht verzweigt. Nach Teil 3 von Lemma 3.9 gibt es u, v, so daß $w = -F\,code(u)F{+}{+}{+}{+}F\,edoc(v)F$ mit $u = u_{i_1} \cdots u_{i_p}$, $v = v_{i_1} \cdots v_{i_p}$, $p \geq 1$. Nach Teil 5 des Lemmas ist $u = v$ und damit $i_1 \cdots i_p$ eine Lösung von PCP. Ist die Antwort positiv, dann verzweigt $d(w)$ für alle $v \in L(G_{PCP})$. Wäre nun PCP dennoch lösbar mit der Lösung $i_1 \cdots i_p$. Dann würde $u_{i_1} \cdots u_{i_p} = u = v = v_{i_1} \cdots v_{i_p}$ gelten und nach Teil 3 und 5 somit $\overline{w} = -F\,code(u)F{+}{+}{+}{+}F\,edoc(v)F \in L(G_{PCP})$ sein und $d(\overline{w})$ nicht verzweigen – im Widerspruch zur Antwort des Algorithmus. Also ist PCP nicht lösbar, und der Algorithmus beantwortet die Frage nach der Lösbarkeit für alle Postschen Korrespondenz-Probleme korrekt. Das aber ist unmöglich, weil bekanntlich die Lösbarkeit Postscher Korrespondenz-Probleme unentscheidbar ist. Also kann es keinen Algorithmus für die Entscheidung des Verzweigungsproblems geben. ∎

Läßt man in Konstruktion 3.8 die drei Regeln in Klammern weg, bleibt eine kontextfreie und zusätzlich sogar lineare Grammatik übrig, die dieselbe Sprache erzeugt. Die Unentscheidbarkeit des Verzweigungsproblems bezüglich der Turtle-Interpretation gilt also selbst für lineare kontextfreie Grammatiken. Für die Schildkröte sind derartige Ergebnisse bisher unbekannt gewesen. Ähnliche Betrachtungen mit ähnlichen Ergebnissen sind allerdings bereits für Kettencode-Bildsprachen angestellt worden. Hinweise darauf finden sich in [DH89]. Wer mehr über Postsche Korrespondenz-Probleme erfahren möchte, kann die schriftlichen Materialien zu *Theoretischer Informatik* I & II zu Rate ziehen oder nahezu jedes Buch über Formale Sprachen (Hopcroft & Ullman, Salomaa, ...).

Kapitel 4

Zelluläre Automaten

Die Konzeption der zellulären Automaten bildet einen der ersten Versuche in der Informatik, parallele Prozesse formal zu fassen. Früh wurde auch erkannt, daß eine geeignete Interpretation zelluläre Automaten zur Erzeugung und Verwandlung von Bildern befähigt.

Mehr über zelluläre Automaten kann in den Büchern von Codd, Preston und Duff sowie Toffoli und Margolus [Cod68, PD84, TM87] nachgelesen werden. Eine kurze, brauchbare Zusammenfassung findet sich bei Peitgen, Jürgens und Saupe [PJS92], von wo auch das in Abschnitt 4.2 diskutierte Beispiel stammt. Weitere Beispiele wie das berühmte *Game of Life* sind dort ebenfalls beschrieben. Zelluläre Automaten werden bisweilen für die Modellierung und Simulation von Prozessen in Natur, Technik und Gesellschaft eingesetzt. Zwei an der Universität Bremen entstandene Beispiele findet man dafür in den Arbeiten von Plath, Schwietering and Swart [PS92a, PS92b].

4.1 Definition und Arbeitsweise zellulärer Automaten

Der Idee nach ist ein zellulärer Automat ein Zellkonglomerat, bei dem jede Zelle einen aktuellen Zustand besitzt und alle Zellen gleichzeitig ihren Zustand in Abhängigkeit von ihrem eigenen Zustand und den aktuellen Zuständen ihrer Nachbarzellen ändern können. Ein bildlicher Effekt entsteht, wenn die Zellen Gebiete in der Ebene sind und die Zustände Farben. Man könnte sich hier beispielsweise vorstellen, daß die Zellen die Pixel eines Bildschirms sind.

4.1 Definition (zellulärer Automat)
Ein *zellulärer Automat* besteht aus

- einer Menge C von *Zellen*;
- einer *Nachbarschaftsfunktion* $N : C \to C^*$, die jeder Zelle eine Sequenz von *Nachbarzellen* zuordnet;
- einer *Automatenzuordnung* A, die jeder Zelle $c \in C$ einen endlichen Automaten A_c zuordnet, wobei A_c eine Zustandsmenge Z_c und eine Zustandsüberführungsfunktion $d_c : Z_c \times Z_{c_1} \times \cdots \times Z_{c_n} \to Z_c$ mit $c_1 \cdots c_n = N(c)$ besitzt, und

- einer *Anfangskonfiguration* gen_0, die jeder Zelle $c \in C$ einen Zustand $gen_0(c) \in Z_c$ zuordnet. Die Anfangskonfiguration wird auch *0-te Generation* genannt.

Ein zellulärer Automat generiert eine Folge von *Konfigurationen* oder *Generationen*

$$gen_0 \to gen_1 \to gen_2 \to \cdots \to gen_i \to gen_{i+1} \to \cdots$$

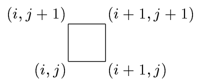
die mit der 0-ten Generation beginnt. Dabei ist gen_{i+1} $(i \geq 0)$ induktiv dadurch definiert, daß es jeder Zelle $c \in C$ mit der Nachbarschaft $N(c) = c_1 \cdots c_n$ den Folgezustand

$$gen_{i+1}(c) = d_c(gen_i(c), gen_i(c_1), \ldots, gen_i(c_n))$$

zuordnet. ◇

Will man zelluläre Automaten implementieren und für die Bilderzeugung nutzen, so empfehlen sich gewisse Einschränkungen und Konventionen. Diese betreffen in erster Linie die betrachteten Nachbarschaften und die endliche Beschreibbarkeit des Automaten. Es ist nämlich nicht sehr leicht, mit Automaten umzugehen, die aus unendlich vielen unterschiedlichen Zellen bestehen. Für den konkreten Umgang mit zellulären Automaten im Bereich der Bilderzeugung einigt man sich daher üblicherweise auf folgende Beschränkungen:

(i) Jede Zelle ist ein Paar $(i, j) \in \mathbb{Z}^2$, welches das Einheitsquadrat in der euklidischen Ebene mit der linken unteren Ecke (i, j) repräsentiert:



Eine Zelle $(i, 0)$ wird auch allein durch $i \in \mathbb{Z}$ dargestellt. Werden nur solche Zellen verwendet, handelt es sich um einen 1-dimensionalen zellulären Automaten, sonst um einen 2-dimensionalen. Im 2-dimensionalen Fall überdecken die Zellen also die gesamte Ebene, im 1-dimensionalen Fall bilden sie ein zweiseitig unendliches Band mit Einheitsbreite. (Wer möchte, kann natürlich auch 3-dimensionale zelluläre Automaten betrachten, in denen die Zellen Einheitswürfel sind.)

(iii) Für alle Zellen wird strukturell dieselbe Nachbarschaft gewählt. Mit anderen Worten, für jede Zelle c mit $N(c) = c_1 \cdots c_n$ und alle $c' \in \mathbb{Z}^2$ gilt $N(c+c') = (c_1+c') \cdots (c_n+c')$. Insbesondere stehen folgende Standardfälle zur Auswahl:

(A) die acht direkten Nachbarn, die eine Kante oder eine Ecke mit der betrachteten Zelle teilen:

2	3	4
1		5
8	7	6

Zur Identifizierung werden die Nachbarn wie gezeigt numeriert;

(B) die vier direkten Nachbarn, die eine Kante mit der betrachteten Zelle teilen:

	N	
W		O
	S	

Sie werden mit den vier Haupthimmelsrichtungen bezeichnet;

(C) drei neben einander liegende Nachbarn aus (A);

(D) der linke und der rechte Nachbar, die mit L bzw. R bezeichnet werden;

L		R

(E) nur der linke Nachbar.

Die letzten beiden Varianten eignen sich offenbar auch für den 1-dimensionalen Fall.

(iv) Alle Zellen erhalten denselben Automaten $A = (Z, d)$ zugeordnet. Die Zustandsüberführung hat damit die Form $d\colon Z \times Z^k \to Z$, wobei $k = 8$ bei Nachbarschaft (A), $k = 4$ bei (B), $k = 3$ bei (C), $k = 2$ bei (D) und $k = 1$ bei (E) ist. Die Generationen werden damit zu Abbildungen $gen\colon C \to Z$.

(v) Generationen werden Bilder, wenn als Zustände Farben gewählt werden. Diese Bilder setzen sich aus farbigen Quadraten zusammen und überdecken die Ebene oder bilden ein Band.

(vi) Zusätzlich nimmt man die Existenz einer speziellen Hintergrundfarbe *white* an, die sich nicht ändert, wenn alle Nachbarn auch *white* sind:

$$d(white, white^k) = white.$$

Außerdem wird verlangt, daß die 0-te Generation nur endlich viele Zellen hat, deren Farbe nicht *white* ist:

$$\{c \in C \mid gen_0(c) \neq white\} \text{ ist endlich.}$$

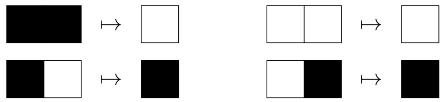
Jede Folgegeneration hat dann auch diese Eigenschaft (Beweis?).

4.2 Beispiel: Der Sierpinski-Automat

Um das Prinzip zu demonstrieren, wird ein zellulärer Automat spezifiziert und untersucht – der sogenannte *Sierpinski-Automat*. Er ist 1-dimensional und hat zwei Farben: $Z = \{black, white\}$. Als Nachbarschaft wird Variante (E) betrachtet, und die Zustandsüberführung ist definiert durch

$$\begin{aligned} d(black, black) &= white \qquad & d(white, white) &= white \\ d(black, white) &= black \qquad & d(white, black) &= black. \end{aligned}$$

Das entspricht einem „exklusiven Oder“ und kann anschaulicher wie folgt dargestellt werden (wobei die linke Seite jeweils die Zelle mit der linken Nachbarzelle darstellt und die rechte Seite den neuen Zustand der Zelle):



Wählt man als 0-te Generation die, in der nur die Zelle 0 den Zustand *black* hat, erhält man die Generationen in Abbildung 4.1. In allen Generationen haben die negativen Zellen

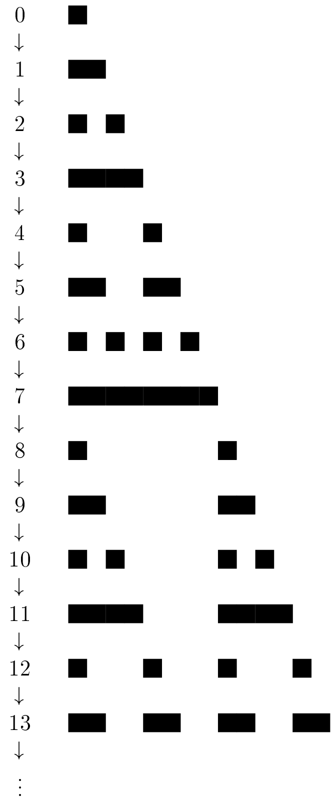
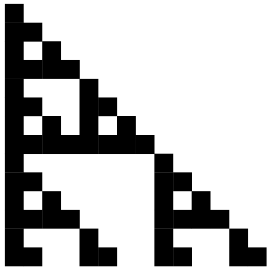


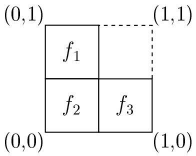
Abbildung 4.1: Generationenfolge des Sierpinski-Automaten

die Farbe *white*, weil sie und ihre linken Nachbarn in vorausgehenden Generationen diese Farbe haben. Die Zelle 0 ist immer *black*, weil sie in jeder vorausgehenden Generation *black* ist und ihr linker Nachbar *white*. In der k-ten Generation ist außerdem die k-te Zelle die größte mit der Farbe *black*, weil in der vorausgehenden Generation die $(k-1)$-te Zelle diese Eigenschaft hat (und somit in der Generation die k-te Zelle *white* ist).

Klebt man die Generationen alle untereinander, entsteht ein zweidimensionales Bild, das schon eher verrät, woher der Automat seinen Namen hat:



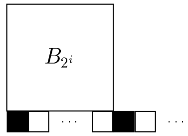
Der Eindruck, daß im Grenzwert eine (unendlich große) Variante des Sierpinski-Dreiecks entstehen wird, ist nicht falsch. Um das einzusehen, sei B_i $(i \geq 1)$ das aus den ersten i Generationen $gen_0, \ldots, gen_{i-1}$ gewonnene Bild, wobei die linke untere Ecke (also $gen_{i-1}(0)$) im Ursprung liegt, und S_i das Bild, das aus B_i mittels Skalierung um den Faktor $1/i$ entsteht. S_i ist also gerade soweit verkleinert, daß es in das Einheitsquadrat paßt. S_1 ist das (ausgefüllte) Einheitsquadrat. Nun betrachte das IFS $F = \{f_1, f_2, f_3\}$



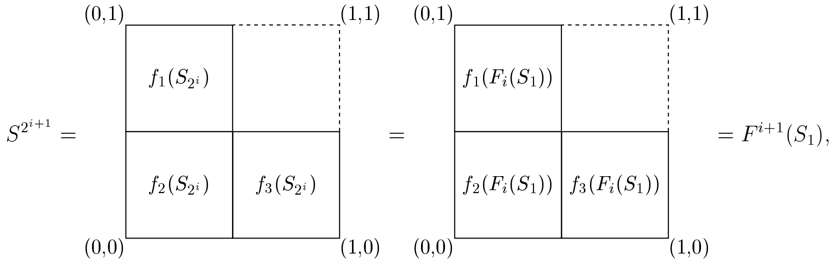
zur Erzeugung der rechtwinkligen Variante des Sierpinski-Dreiecks.

Behauptung $S_{2^i} = F^i(S_1)$ für alle $i \in \mathbb{N}$.

Beweisen kann man dies per Induktion über i. Für $i = 0$ gilt die Behauptung, denn $S_{2^0} = S_1 = F^0(S_1)$. Als Induktionsvoraussetzung nehme man nun an, die Behauptung gelte für ein $i \in \mathbb{N}$, d.h. $S_{2^i} = F^i(S_1)$. Dann gilt insbesondere $gen_{2^i-1}(j) = black$ für $0 \leq j < 2^i$ (und $gen_{2^i-1}(j) = white$ für alle übrigen $j \in \mathbb{Z}$), da die untere Kante von $F^i(S_0)$ ausschließlich aus schwarzen Quadraten besteht. Daraus folgt nach Definition des Automaten, dass $gen_{2^i}(j)$ genau dann *black* ist, wenn $j = 0$ oder $j = 2^i$. B_{2^i+1} sieht also so aus:



Da außer ihrem eigenen nur der Zustand der linken Nachbarzelle einen Einfluß auf den Zustand einer Zelle der nächsten Generation hat, ergibt sich daraus die Gleichung $gen_{2^i+j}(j) = gen_j(j)$ für $0 \leq j < 2^i$. Der untere linke Quadrant von $B_{2^{i+1}}$ ist also jedenfalls identisch zu B_{2^i}. Damit ist aber auch $gen_{2^i}(2^i - 1) = \cdots = gen_{2^{i+1}-2}(2^i - 1) = white$. Mit anderen Worten, die Zellen links neben dem rechten Quadranten sind bis zur vorletzten Generation *white*. Also entwickelt sich dieser Quadrant unbeeinflußt von dem links neben ihm liegenden ebenfalls wie B_{2^i} und wir erhalten unter Benutzung der Induktionsvoraussetzung



was zu zeigen war.

4.3 Zusammenhang zwischen dem Sierpinski-Automaten und dem Pascalschen Dreieck

Wie sich Eigenschaften zellulärer Automaten beweisen lassen, kann auch am Zusammenhang zwischen dem Sierpinski-Automaten und dem Pascalschen Dreieck gezeigt werden. Letzteres ist eine unendliche Dreiecksmatrix, deren Zeilen und Spalten mit den natürlichen Zahlen indiziert sind. Der Eintrag in der n-ten Zeile und k-ten Spalte $(0 \leq k \leq n)$ wird mit $\binom{n}{k}$ bezeichnet (gelesen: n über k) und nach folgendem Verfahren berechnet:

(i) $\binom{n}{0} = \binom{n}{n} = 1$ für alle $n \in \mathbb{N}$

(ii) $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$ für alle $n, k \in \mathbb{N}$ mit $0 < k < n$.

Die Signifikanz des Pascalschen Dreiecks besteht darin, daß die Einträge gerade die Koeffizienten der binomischen Formel sind:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

Bei scharfem Hinsehen (Intuition, Einsicht, Überlegung, Glück) kann vermutet werden, daß die k-te Zelle in der n-ten Generation des Sierpinski-Automaten genau dann *black* ist, wenn $\binom{n}{k}$ ungerade ist. Zur Schreibvereinfachung sei im folgenden $\begin{bmatrix} n \\ k \end{bmatrix} = \binom{n}{k} \mod 2$.

Behauptung Für alle $n, k \in \mathbb{N}$ mit $0 \leq k \leq n$ gilt:

$$gen_n(k) = black \quad \text{g.d.w.} \quad \begin{bmatrix} n \\ k \end{bmatrix} = 1$$

Der Beweis erfolgt per Induktion über n.

I.A.: $n = 0$. Nach Definition ist $gen_0(0) = black$ und $\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \binom{0}{0} = 1$.

I.V.: Die Behauptung gelte für n.

I.S.: Nach Definition gilt $\begin{bmatrix} n+1 \\ k \end{bmatrix} = \binom{n}{k-1} + \binom{n}{k} \mod 2$. Bekannten arithmetischen Gesetzen zufolge ist eine Summe genau dann ungerade, wenn ein Summand gerade und der andere ungerade ist. Also ergibt sich

$$\begin{aligned} \begin{bmatrix} n+1 \\ k \end{bmatrix} = 1 \quad & \Longleftrightarrow \quad \begin{bmatrix} n \\ k \end{bmatrix} \neq \begin{bmatrix} n \\ k-1 \end{bmatrix} \\ & \Longleftrightarrow \quad gen_n(k) \neq gen_n(k-1) \\ & \Longleftrightarrow \quad gen_{n+1}(k) = black, \end{aligned}$$

wobei für die zweite Äquivalenz die Induktionsvoraussetzung und für die dritte die Definition des Sierpinski-Automaten verwendet wurde.

Literaturverzeichnis

[Cod68] E.-F. Codd. *Cellular Automata*. Academic Press, 1968.

[DH89] Jürgen Dassow and Friedhelm Hinz. Kettenkode-Bildsprachen. *Wissenschaftliche Zeitschrift der TU Magdeburg*, 33(6):1–12, 1989.

[Edg90] Gerald A. Edgar. *Measure, Topology, and Fractal Geometry*. Springer, New York, 1990.

[Fre61] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Trans. Electron. Comput.*, 10:260–268, 1961.

[Fre74] H. Freeman. Computer processing of line-drawing images. *Computer Surveys*, 6:57–97, 1974.

[MRW82] Hermann A. Maurer, Grzegorz Rozenberg, and Emo Welzl. Using string languages to describe picture languages. *Information and Control*, 54:155–185, 1982.

[PD84] K. Preston and M. Duff. *Modern Cellular Automata, Theory and Application*. Plenum Press, New York, 1984.

[PJS92] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals. New Frontiers of Science*. Springer-Verlag, New York, 1992.

[PL90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[PS92a] P.J. Plath and J. Schwietering. Improbable events in deterministically growing patterns. In J.L. Encarnaçao, H.-O. Peitgen, G. Sakus, and G. Englert, editors, *Fractal Geometry and Computer Graphics*, pages 162–172. Springer, Berlin-Heidelberg, 1992.

[PS92b] P.J. Plath and E.-J. Swart. Simulation of individual behaviour. In J.L. Encarnaçao, H.-O. Peitgen, G. Sakus, and G. Englert, editors, *Fractal Geometry and Computer Graphics*, pages 144–161. Springer, Berlin-Heidelberg, 1992.

[TM87] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines*. MIT Press, Cambridge, Massachusetts, 1987.