

4-Wege-Automaten und Tiling-Systeme

Angelika Schwarz

Seminar über Angewandte Automatentheorie

<http://www.automata.rwth-aachen.de/teaching/ws1112/seminar.en>

Inhaltsverzeichnis

1	Einleitung	1
2	Definitionen und Modelle	1
2.1	Grundlagen	1
2.2	Reguläre Ausdrücke	3
2.3	On-line tessellation automata	6
2.4	Logische Formeln	7
2.5	Tiling-Systeme	8
2.6	Domino-Systeme	10
3	Abschlusseigenschaften	12
4	Äquivalenzaussagen	15
4.1	$L(2OTA) = L(TS)$	15
4.2	$L(DS) = L(TS)$	17
4.3	$L(EMSO) = L(TS)$	19
5	Unentscheidbarkeit	20
6	Fazit	22

1 Einleitung

Reguläre eindimensionale Sprachen können auf viele verschiedene Arten dargestellt werden, wie z.B. durch endliche Automaten oder reguläre Ausdrücke. In [GR97] werden Modelle für die Erweiterung auf zweidimensionale Sprachen vorgestellt. Es wird gezeigt, in welchem Verhältnis diese Modelle zueinander stehen, ob sich die Abschlusseigenschaften wie im eindimensionalen Fall übertragen lassen und wie sich die Erweiterung auf zwei Dimensionen auf die Berechenbarkeit auswirkt.

2 Definitionen und Modelle

2.1 Grundlagen

Im Folgenden werden Begriffe und Notationen für zweidimensionale Sprachen eingeführt.

Definition Ein *picture* ist eine Matrix mit Einträgen aus Σ . Mit Σ^{**} wird die Menge aller pictures über Σ bezeichnet. Eindimensionale Sprachen, *string languages*, sind eine Teilmenge von Σ^{**} . Die $0 \times n$ bzw. $n \times 0$ Matrix heißt *leeres picture* und wird mit λ bezeichnet.

Bemerkung In [GR97] wird das *leere picture* als 0×0 -Matrix definiert; pictures der Größe $0 \times n$ bzw. $n \times 0$ mit $n > 0$ sind nicht definiert. Diese Definition wurde abgeändert, um eine sinnvollere Anwendung von Zeilen- und Spaltenkonkatenation eines pictures

mit λ zu erreichen, da nun die Bedingung an Zeilen- und Spaltenanzahl auch für die Konkatenationsoperationen mit λ erfüllt ist.

Für ein picture $p \in \Sigma^{**}$ bezeichnet $l_1(p)$ die Anzahl der Zeilen, $l_2(p)$ die Anzahl der Spalten und $p(i, j)$ mit $1 \leq i \leq l_1(p), 1 \leq j \leq l_2(p)$ den Eintrag aus Σ , der in p an Position (i, j) steht.

Definition Ein picture $p \in \Sigma^{**}$ mit einer Rahmenbegrenzung und Begrenzungssymbol $\# \notin \Sigma$ wird mit $\hat{p}$ bezeichnet.

$$\hat{p} = \begin{array}{|ccc|} \hline \# & \# & \# \\ \hline \# & p & \# \\ \hline \# & \# & \# \\ \hline \end{array}$$

Im Folgenden werden Operationen für zweidimensionale Sprachen vorgestellt.

Definition Es gibt zwei Konkatenationsoperationen für zweidimensionale Sprachen:

- *Spaltenkonkatenation*: Falls pictures p und q bezüglich ihrer Zeilenanzahl übereinstimmen, ist

$$p \oplus q = \begin{array}{|ccc|ccc|} \hline p_{11} & \cdots & p_{1n} & q_{11} & \cdots & q_{1n'} \\ \hline \vdots & & \vdots & \vdots & & \vdots \\ \hline p_{m1} & \cdots & p_{mn} & q_{m1} & \cdots & q_{mn'} \\ \hline \end{array}$$

- *Zeilenkonkatenation*: Falls die Spaltenanzahl übereinstimmt, ist:

$$p \ominus q = \begin{array}{|ccc|} \hline p_{11} & \cdots & p_{1n} \\ \hline \vdots & & \vdots \\ \hline p_{m1} & \cdots & p_{mn} \\ \hline q_{11} & \cdots & q_{1n} \\ \hline \vdots & & \vdots \\ \hline q_{m'1} & \cdots & q_{m'n} \\ \hline \end{array}$$

Für $m = 1$ entspricht die Spaltenkonkatenation der Konkatenation der eindimensionalen Sprachen. Analog zur Konkatenation mit ϵ im eindimensionalen Fall sind auch die Konkatenationsoperationen mit dem leeren picture definiert.

$$\lambda \oplus p = p = p \oplus \lambda$$

$$\lambda \ominus p = p = p \ominus \lambda$$

Durch Anwenden von Spalten- und Zeilenkonkatenation auf Sprachen können neue Sprachen definiert werden.

Definition Für zwei Sprachen $L_1, L_2 \subseteq \Sigma^{**}$ können durch Spalten- bzw. Zeilenkonkatenation definiert werden

- $L_1 \oplus L_2 = \{p \oplus q \mid p \in L_1 \text{ und } q \in L_2\}$
- $L_1 \ominus L_2 = \{p \ominus q \mid p \in L_1 \text{ und } q \in L_2\}$

Durch die wiederholte Anwendung der Konkatenationsoperationen kann ein Pendant zum Kleene-Stern definiert werden.

Definition Für $L \subseteq \Sigma^{**}$ ist :

- der Abschluss unter Spaltenkonkatenation definiert als

$$L^{*\oplus} := \bigcup_{i \geq 0} L^{i\oplus}$$
mit $L^{0\oplus} := \lambda$, $L^{1\oplus} := L$ und $L^{n\oplus} := L \oplus L^{(n-1)\oplus}$
- der Abschluss unter Zeilenkonkatenation definiert als

$$L^{*\ominus} := \bigcup_{i \geq 0} L^{i\ominus}$$
mit $L^{0\ominus} := \lambda$, $L^{1\ominus} := L$ und $L^{n\ominus} := L \ominus L^{(n-1)\ominus}$

Definition Eine *Projektion* ist eine Abbildung $\pi : \Gamma^{**} \rightarrow \Sigma^{**}$. Ein picture $p \in \Gamma^{**}$ wird Eintrag für Eintrag abgebildet: $p'(i, j) = \pi(p(i, j))$ für $1 \leq i \leq l_1(p)$, $1 \leq j \leq l_2(p)$.

Definition Ein *Block* der Größe (h, w) ist ein Ausschnitt eines pictures der Größe (m, n) mit $h \leq m, w \leq n$. $B_{h,w}(p)$ bezeichnet die Menge aller Blöcke von p der Größe (h, w) .

Im Folgenden werden verschiedene Modelle zur Beschreibung zweidimensionaler Sprachen vorgestellt.

2.2 Reguläre Ausdrücke

Reguläre Ausdrücke für zweidimensionale Sprachen sind wie im eindimensionalen Fall rekursiv definiert. Als Operationen sind $\{\oplus, \ominus, *^\oplus, *^\ominus, \cup, \cap, \neg\}$ möglich.

Definition Ein *regulärer Ausdruck* für zweidimensionale Sprachen über dem Alphabet Σ lässt sich aufbauen als:

- $\emptyset$ und jedes $a \in \Sigma$ sind reguläre Ausdrücke
- Für zwei reguläre Ausdrücke α, β sind ebenfalls regulär $\alpha \otimes \beta$ mit $\otimes \in \{\oplus, \ominus, \cup, \cap\}$, $\bar{\alpha}$ und $(\alpha)^{*\oplus}$ bzw. $(\alpha)^{*\ominus}$

Definition Eine Sprache L ist eine Projektion eines komplementfreien regulären Ausdrucks, $L \in L(PCFRE)$, wenn gilt:

- L ist Projektion einer Sprache L' : $\pi(L') = L$
- L' ist als regulärer Ausdruck über den Operationen $\{\oplus, \ominus, *^\oplus, *^\ominus, \cup, \cap\}$ darstellbar.

Als durchgängiges Beispiel dient die Schachbrettsprache. Damit sind alle quadratischen, nichtleeren pictures gemeint, die ein Schachbrettmuster haben und deren Eintrag oben

links ein a ist.

$$L_{\text{Schachbrett}} = \left\{ p \in \{a, b\}^{**} \mid \begin{array}{l} p \neq \lambda, l_1(p) = l_2(p), p(1, 1) = a, \\ p(i, j) \neq p(i, j+1) \text{ für } 1 \leq i \leq l_1(p), 1 \leq j < l_2(p) \\ p(i, j) \neq p(i+1, j) \text{ für } 1 \leq i < l_1(p), 1 \leq j \leq l_2(p) \end{array} \right\}$$

$$= \left\{ \boxed{a}, \begin{array}{|c|c|} \hline a & b \\ \hline b & a \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline a & b & a \\ \hline b & a & b \\ \hline a & b & a \\ \hline \end{array}, \dots \right\}$$

Beispiel Schachbretter folgender Form werden zunächst durch einen komplementfreien regulären Ausdruck beschrieben und anschließend auf das Alphabet $\{a, b\}$ abgebildet.

a_0	b_2	a_2	b_2	a_2	b_2	a_2
b_1	a_0	b_2	a_2	b_2	a_2	b_2
a_1	b_1	a_0	b_2	a_2	b_2	a_2
b_1	a_1	b_1	a_0	b_2	a_2	b_2
a_1	b_1	a_1	b_1	a_0	b_2	a_2
b_1	a_1	b_1	a_1	b_1	a_0	b_2
a_1	b_1	a_1	b_1	a_1	b_1	a_0

Die Diagonale wird mit einem speziellen Zeichen fixiert, für die obere Dreiecksmatrix und die untere Dreiecksmatrix werden separate Alphabetsymbole verwendet. Für Schachbretter gerader und ungerader Seitenlänge werden disjunkte Alphabetsymbole verwendet. Betrachtet man ein Schachbrett spaltenweise, so lassen sich für die Spalten ein regulärer Ausdruck β_v^u für ungerade bzw. β_v^g für gerade Seitenlängen angeben. Analog verfährt man für die Zeilen und erhält β_h^u für ungerade bzw. β_h^g für gerade Seitenlängen.

$$\beta_v^u = \underbrace{((a_2 \ominus b_2)^{* \ominus}) \ominus a_0 \ominus ((b_1 \ominus a_1)^{* \ominus})}_{\text{gerade Spaltennummer}} \cup \underbrace{b_2 \ominus ((a_2 \ominus b_2)^{* \ominus}) \ominus a_0 \ominus ((b_1 \ominus a_1)^{* \ominus}) \ominus b_1}_{\text{ungerade Spaltennummer}}$$

$$\beta_h^u = \underbrace{((a_1 \oplus b_1)^{* \oplus}) \oplus a_0 \oplus ((b_2 \oplus a_2)^{* \oplus})}_{\text{ungerade Zeilennummer}} \cup \underbrace{b_1 \oplus ((a_1 \oplus b_1)^{* \oplus}) \oplus a_0 \oplus ((b_2 \oplus a_2)^{* \oplus}) \oplus b_2}_{\text{gerade Zeilennummer}}$$

$$\beta_v^g = ((a'_2 \ominus b'_2)^{* \ominus}) \ominus a'_0 \ominus ((b'_1 \ominus a'_1)^{* \ominus}) \ominus b'_1 \cup b'_2 \ominus ((a'_2 \ominus b'_2)^{* \ominus}) \ominus a'_0 \ominus ((b'_1 \ominus a'_1)^{* \ominus})$$

$$\beta_h^g = ((a'_1 \oplus b'_1)^{* \oplus}) \oplus a'_0 \oplus ((b'_2 \oplus a'_2)^{* \oplus}) \oplus b'_2 \cup b'_1 \oplus ((a'_1 \oplus b'_1)^{* \oplus}) \oplus a'_0 \oplus ((b'_2 \oplus a'_2)^{* \oplus})$$

Die obigen Gleichungen beschreiben Zeilen oder Spalten, also Strings. Für ein zweidimensionales picture wird die Spaltenkonkatenation auf Zeilen bzw. die Zeilenkonkatenation auf Spalten angewendet. Ein gültiges picture muss sowohl die Bedingung an die Zeilen als auch an die Spalten erfüllen.

$$L = \underbrace{\left((\beta_v^u)^{* \oplus} \right) \cap \left((\beta_h^u)^{* \ominus} \right)}_{\text{pictures ungerader Seitenlänge}} \cup \underbrace{\left((\beta_v^g)^{* \oplus} \right) \cap \left((\beta_h^g)^{* \ominus} \right)}_{\text{pictures gerader Seitenlänge}}$$

$$\pi : \{a_0, a_1, a_2, a'_0, a'_1, a'_2, b_1, b_2, b'_1, b'_2\} \rightarrow \{a, b\}$$

$$\pi(a_0) = \pi(a_1) = \pi(a_2) = \pi(a'_0) = \pi(a'_1) = \pi(a'_2) = a$$

$$\pi(b_1) = \pi(b_2) = \pi(b'_1) = \pi(b'_2) = b$$

Die Anwendung der Projektion liefert $\pi(L) = L_{\text{Schachbrett}}$.

2.3 On-line tessellation automata

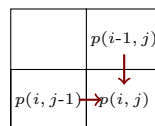
Ein NEA liest ein Eingabewort Zeichen für Zeichen und befindet sich beim Lesen jeden Zeichens in einem Zustand. So gesehen wird jedem Zeichen des Inputs ein Zustand zugewiesen. Das folgende Automatenmodell, der 2OTA, ist eine Verallgemeinerung eines NEAs auf zweidimensionale Sprachen. Ein 2OTA ordnet jeder Position eines pictures einen Zustand zu. Die schrittweise Vorgehensweise des NEAs, eine Eingabe Zeichen für Zeichen zu lesen, übersetzt sich für den 2OTA in diagonalenweises Scannen eines pictures. In dieser Semantik akzeptiert ein NEA ein Wort, wenn dem letzten Zeichen der Eingabe ein Endzustand zugewiesen wird. Analog dazu akzeptiert ein 2OTA, wenn das letzte gelesene Zeichen - der untere rechte Rand eines pictures - mit einem Endzustand identifiziert wird. Genauso wie ein NEA verwirft auch der 2OTA eine Eingabe, wenn während eines Laufs keine weitere Transition anwendbar ist.

Definition Ein *nichtdeterministischer 2OTA* (*two-dimensional on-line tessellation automaton*) ist ein Automat $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ mit

- Σ Eingabealphabet
- Q Zustandsmenge
- $I \subseteq Q$ eine Menge von Startzuständen
- $F \subseteq Q$ eine Menge von Endzuständen
- $\delta : Q \times Q \times \Sigma \rightarrow 2^Q$ als Zustandsübergangsfunktion

Ein *Lauf* eines 2OTAs $\mathcal{A}$ auf einem picture p ordnet jeder Position (i, j) von p folgendermaßen einen Zustand zu:

- Der Zustand von (i, j) hängt ab von den Zuständen an den Positionen $(i-1, j)$ und $(i, j-1)$ sowie dem Symbol $p(i, j)$.

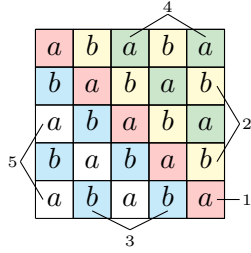


- Für $t = 0$ werden die erste Spalte und die erste Zeile von $\hat{p}$ mit dem Startzustand identifiziert.
- Für $t = k$ werden simultan allen Position (i, j) mit $i + j - 1 = k$ entsprechend δ je ein Zustand zugewiesen.
- Der Lauf ist akzeptierend, wenn $(l_1(p), l_2(p))$ mit einem Endzustand identifiziert wird.

Ein picture wird in einer fest vorgegebenen Reihenfolge (nämlich von der Ecke oben links Diagonale für Diagonale zur Ecke unten rechts) durchlaufen. Es werden an einer Position gleichzeitig die Übergangsfunktion ausgewertet und nichtdeterministisch dieser

Position „der richtige“ Zustand zugeordnet. Ein picture gehört zur Sprache des 2OTAs, wenn der unteren rechten Ecke des pictures ein Endzustand zugeordnet wird.

Beispiel Ein 2OTA, der die Schachbrettsprache erkennt, ist gegeben durch:



- $Q = \{0, 1, 2, 3, 4, 5\}$
- $I = \{0\}$
- $F = \{1\}$
- $\delta :$

	0	1	2	3	4	5
	a b	a b	a b	a b	a b	a b
0	1	3		5		3
1		2			2	
2	4		4			
3			1	5		
4		2			2	
5			3			3

Die im Bild nicht eingezeichneten #-Symbole werden mit dem Zustand 0, dem Startzustand, identifiziert. Nach [PO04] ist δ folgendermaßen zu lesen: Einer Transition $\delta(q_u, q_l, a) = q$ entspricht dieser Eintrag in der Transitionstabelle:

	q_u
	a
q_l	q

2.4 Logische Formeln

Definition Ein picture p der Größe (m, n) kann mithilfe der Signatur $\underline{p} = (dom(p), S_1, S_2, (P_a)_{a \in \Sigma})$ dargestellt werden, wobei:

- $dom(p) = \{1, \dots, l_1(p)\} \times \{1, \dots, l_2(p)\}$
- S_1, S_2 sind binäre Nachfolgerrelationen. Für zwei Positionen aus $dom(p)$ gilt $(i, j)S_1(i+1, j)$ und $(i, j)S_2(i, j+1)$ für $1 \leq i \leq m, 1 \leq j \leq n$
- $P_a = \{(i, j) \mid p(i, j) = a\}$ für $a \in \Sigma$

Ein picture p erfüllt eine MSO-Formel $\varphi(X_1, \dots, X_n)$ mit den freien Variablen $X_1, \dots, X_n$, wenn für $Q_1, \dots, Q_n \subseteq dom(p)$

$$(\underline{p}, Q_1, \dots, Q_n) \models \varphi(X_1, \dots, X_n)$$

Wenn φ ein Satz ist

$$\underline{p} \models \varphi$$

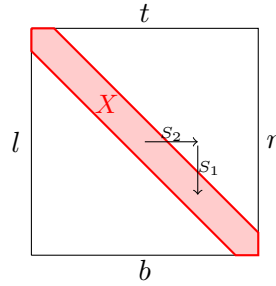
Die Sprache, die durch einen Satz φ beschrieben wird, umfasst alle pictures, die φ erfüllen. $L(\varphi) = \{p \in \Sigma^{**} \mid \underline{p} \models \varphi\}$

Beispiel Die Schachbrettsprache kann durch eine EMSO-Formel definiert werden.

Die Ränder eines pictures (t: oben, l: links, tl: oben links, usw.) sind darstellbar mittels FO-Formeln.

$$\begin{aligned}\varphi_t(x) &= \neg \exists y (y S_1 x) \\ \varphi_l(x) &= \neg \exists y (y S_2 x) \\ \varphi_{tl}(x) &= \varphi_t(x) \wedge \varphi_l(x) \\ &\vdots\end{aligned}$$

Ein picture ist quadratisch, wenn man entlang der Diagonalen nach unten läuft und die letzte so erreichbare Position weder einen rechten noch einen unteren Nachfolger hat.



$$\begin{aligned}\varphi = & \exists X (\exists x (\varphi_{tl}(x) \wedge X(x)) \wedge \forall x (X(x) \rightarrow P_a(x)) & // X \text{ bezeichnet die Positionen auf der Diagonale} \\ & \wedge \forall x \forall y \forall z (X(x) \wedge x S_1 y \wedge y S_2 z \rightarrow X(z)) & // \text{entlang Diagonale mit „Treppenstufen“ laufen} \\ & \wedge \forall ((\varphi_b(x) \vee \varphi_r(x)) \rightarrow (\neg X(x) \vee \varphi_{br}(x))) & // \text{Bedingung, dass picture quadratisch ist} \\ & \wedge \forall x \exists y (P_a(x) \wedge \neg \varphi_r(x) \rightarrow x S_2 y \wedge P_b(y)) & // \text{„Schachbrettmuster“, d.h. } a \text{ s und } b \text{ s wechseln} \\ & \wedge \forall x \exists y (P_a(x) \wedge \neg \varphi_b(x) \rightarrow x S_1 y \wedge P_b(y)) & \text{ sich ab} \\ & \wedge \forall x \exists y (P_b(x) \wedge \neg \varphi_r(x) \rightarrow x S_2 y \wedge P_a(y)) \\ & \wedge \forall x \exists y (P_b(x) \wedge \neg \varphi_b(x) \rightarrow x S_1 y \wedge P_a(y))\end{aligned}$$

2.5 Tiling-Systeme

Für das Modell eines Tiling-Systems werden einige weitere Definitionen benötigt.

Definition Ein *tile* ist ein Block der Größe $(2, 2)$.

Definition Eine Sprache $L \subseteq \Gamma^{**}$ heißt *lokal*, falls es eine endliche Menge Θ von tiles gibt, so dass $L = \{p \in \Gamma^{**} \mid B_{2,2}(\hat{p}) \subseteq \Theta\}$. LOC bezeichnet die Menge aller lokalen Sprachen.

λ wird mit

#	#
#	#

 identifiziert.

Bemerkung Die Schachbrettsprache ist keine lokale Sprache.

Beweis Angenommen, $L_{\text{Schachbrett}}$ wäre eine lokale Sprache. Da ein gültiges Schachbrett

durch

a	b	a
b	a	b
a	b	a

 gegeben ist, muss Θ insbesondere folgende tiles enthalten:

#	#	#	#	#	#	#	a	#	b	#	a	a	b	b	a	a	#
#	a	a	b	b	a	a	#	#	b	#	a	#	#	#	#	#	#

a	$\#$
b	$\#$

,

b	$\#$
a	$\#$

,

b	a
a	b

,

a	b
b	a

Mit diesen tiles lässt sich auch ein picture $\hat{p} =$

#	#	#	#	#
#	a	b	a	#
#	#	#	#	#

 konstruieren, aber

$p \notin L_{\text{Schachbrett}}$.

Definition Ein *Tiling-System* $\mathcal{T}$ ist ein Quadrupel $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ mit

- Σ, Γ zwei Alphabete
- Θ eine Menge von tiles über $\Gamma \cup \{\#\}$
- $\pi : \Gamma \rightarrow \Sigma$ eine Projektion

Ein Tiling-System erkennt eine Sprache $L \subseteq \Sigma^{**}$, falls $L = \pi(L(\Theta))$.

$L(TS) = \{L \mid L \text{ ist durch ein Tiling-System erkennbar} \}$

Tiling-Systeme sind eine Art zweidimensionaler NEA. Denn wenn man das lokale Alphabet als $\Gamma = \Sigma \times Q$ auffasst und die Projektion als $\pi(a, q) = a, a \in \Sigma, q \in Q$ wählt, ist ein Tiling-System (vgl. Kapitel 4.1) eine Fortsetzung von NEAs. Ein Tiling-System ist also implizit nichtdeterministisch. Ein Tiling-System erkennt insbesondere eindimensionale reguläre Sprachen.

Beispiel Das Beispiel zeigt ein Tiling-System, das die Schachbrettsprache erkennt. Mithilfe der lokalen Sprache wird die quadratische Struktur vorgeschrieben, indem die Diagonale durch ein zusätzliches Alphabetsymbol a_0 fixiert wird. Weiter wird das abwechselnde Auftreten der Alphabetsymbole vorgeschrieben. Das korrekte Alphabet $\Sigma = \{a, b\}$ wird durch die Projektion erreicht.

Die lokale Sprache ist durch diese tiles gegeben:

$$\Theta = \left\{ \begin{array}{ccccccc} \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & a_0 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & \# \\ \hline a_0 & b_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & \# \\ \hline b_1 & a_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & \# \\ \hline a_1 & b_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & \# \\ \hline a_1 & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & \# \\ \hline b_1 & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & \# \\ \hline a_0 & \# \\ \hline \end{array}, \\ \begin{array}{|c|c|} \hline \# & a_0 \\ \hline \# & b_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & b_1 \\ \hline \# & a_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & a_1 \\ \hline \# & b_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & a_1 \\ \hline \# & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & b_1 \\ \hline \# & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline \# & a_0 \\ \hline \# & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline a_1 & b_1 \\ \hline \# & \# \\ \hline \end{array}, \\ \begin{array}{|c|c|} \hline b_1 & a_1 \\ \hline \# & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline a_0 & \# \\ \hline \# & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline b_1 & a_0 \\ \hline \# & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline a_1 & \# \\ \hline b_1 & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline b_1 & \# \\ \hline a_1 & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline b_1 & \# \\ \hline a_0 & \# \\ \hline \end{array}, & \begin{array}{|c|c|} \hline a_0 & b_1 \\ \hline b_1 & a_0 \\ \hline \end{array}, \\ \begin{array}{|c|c|} \hline b_1 & a_1 \\ \hline a_0 & b_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline b_1 & a_1 \\ \hline a_1 & b_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline a_1 & b_1 \\ \hline b_1 & a_1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline b_1 & a_0 \\ \hline a_1 & b_1 \\ \hline \end{array} \end{array} \right\}$$

Damit werden pictures folgender Form beschrieben:

$$\begin{array}{cccccccc} \# & \# & \# & \# & \dots & \# & \# & \# \\ \# & a_0 & b_1 & a_1 & \dots & & & \# \\ \# & b_1 & a_0 & b_1 & & & & \# \\ \# & a_1 & b_1 & a_0 & & & & \# \\ \vdots & \vdots & & & \ddots & & & \vdots \\ \# & & & & & & b_1 & \# \\ \# & & & & & & b_1 & a_0 & \# \\ \# & \# & \# & \# & \dots & \# & \# & \# \end{array}$$

Projektion:

$$\begin{array}{ccc} \pi : \Gamma & \rightarrow & \Sigma \\ \parallel & & \parallel \\ \{a_0, a_1, b_1\} & & \{a, b\} \end{array} \quad \text{mit} \quad \begin{array}{l} \pi(a_0) = \pi(a_1) = a \\ \pi(b_1) = b \end{array}$$

2.6 Domino-Systeme

Domino-Systeme sind ähnlich wie ein Tiling-System über einer lokalen Sprache und einer Projektion aufgebaut. Die lokale Sprache eines Domino-Systems setzt sich aus Blöcken der Größe $(1, 2)$ (horizontales Domino) bzw. $(2, 1)$ (vertikales Domino) zusammen und heißt *hv-lokal*.

Definition Ein *Domino* ist ein Block der Größe $(1, 2)$ bzw. $(2, 1)$.

Definition Eine Sprache $L \subseteq \Gamma^{**}$ heißt *hv-lokal*, falls es eine endliche Menge Δ von Dominos über $\Gamma \cup \{\#\}$ gibt, so dass $L = \{p \in \Gamma^{**} \mid (B_{1,2}(\hat{p}) \cup B_{2,1}(\hat{p})) \subseteq \Delta\}$.

$$\begin{array}{|c|c|} \hline \# & \# \\ \hline \end{array} \in L \text{ und } \begin{array}{|c|} \hline \# \\ \hline \# \\ \hline \end{array} \in L \Leftrightarrow \lambda \in L.$$

Lemma 2.1 Die Menge der hv-lokalen Sprachen ist eine echte Teilmenge der lokalen Sprachen.

Beweis „ $\subseteq$ “: Für die Inklusion $\text{hv-lokal} \subseteq \text{LOC}$ ist zu zeigen, dass zu einer hv-lokalen Sprache $L \subseteq \Gamma^{**}$, die durch eine Menge von Δ von Dominos dargestellt werden kann, eine lokale Sprache, d.h. eine Menge Θ von tiles, konstruiert werden kann mit $L = L(\Gamma) = L(\Theta)$. Im Beweis von $L(DS) \subseteq L(TS)$ im Kapitel 4.2 wird gezeigt, wie zu Δ entsprechende tiles definiert werden können.

„ $\not\subseteq$ “: Die folgende Sprache, die aus genau einem picture besteht, ist durch eine Menge Θ von tiles erkennbar, aber nicht durch Dominos.

$$L = \left\{ \hat{p} = \begin{array}{|c|c|c|c|} \hline \# & \# & \# & \# \\ \hline \# & a & b & \# \\ \hline \# & b & a & \# \\ \hline \# & \# & \# & \# \\ \hline \end{array} \right\}$$

$$\Theta = \left\{ \begin{array}{|c|c|}, \begin{array}{|c|c|}, \begin{array}{|c|c|}, \begin{array}{|c|c|}, \begin{array}{|c|c|}, \begin{array}{|c|c|} \\ \hline \# & \# \\ \hline \# & a \end{array}, \begin{array}{|c|c|} \\ \hline \# & a \\ \hline a & b \end{array}, \begin{array}{|c|c|} \\ \hline \# & \# \\ \hline b & \# \end{array}, \begin{array}{|c|c|} \\ \hline \# & a \\ \hline \# & b \end{array}, \begin{array}{|c|c|} \\ \hline a & b \\ \hline b & a \end{array}, \\ \hline \begin{array}{|c|c|} \\ \hline b & \# \\ \hline a & \# \end{array}, \begin{array}{|c|c|} \\ \hline \# & b \\ \hline \# & \# \end{array}, \begin{array}{|c|c|} \\ \hline b & a \\ \hline \# & \# \end{array}, \begin{array}{|c|c|} \\ \hline a & \# \\ \hline \# & \# \end{array} \end{array} \right\}$$

Es gilt $L = L(\Theta)$. Insbesondere können durch die tiles keine weiteren pictures erzeugt werden. Da alle $(1, 2)$ - und $(2, 1)$ -Blöcke von $\hat{p}$ in der Menge von Dominos enthalten sein müssen, ergibt sich Δ als:

$$\Delta = \left\{ \begin{array}{|c|c|}, \begin{array}{|c|} \\ \hline a_1 \\ \hline a_2 \end{array}, \begin{array}{|c|c|} \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|} \\ \hline \# \\ \hline \# \end{array} \mid a_1, a_2 \in \{a, b, \#\}, a_1 \neq a_2, \right\}$$

Dann kann z.B. $\hat{p}' = \begin{array}{|c|c|c|c|c|} \hline \# & \# & \# & \# & \# \\ \hline \# & a & b & a & \# \\ \hline \# & \# & \# & \# & \# \\ \hline \end{array}$ durch Dominos in Δ konstruiert werden.

Somit ist $p' \in L(\Delta)$, aber $p' \notin L$. Also ist L eine lokale, aber keine hv-lokale Sprache.

Definition Ein *Domino-System* $\mathcal{D}$ ist ein Quadrupel $\mathcal{D} = (\Sigma, \Gamma, \Delta, \pi)$ mit

- Σ, Γ zwei Alphabete
- Δ eine Menge von Dominos über $\Gamma \cup \{\#\}$
- $\pi : \Gamma \rightarrow \Sigma$ eine Projektion

Ein Domino-System erkennt eine Sprache $L \subseteq \Sigma^{**}$, falls $L = \pi(L(\Gamma))$.

$$L(DS) = \{L \mid L \text{ ist durch ein Domino-System erkennbar} \}$$

Im Gegensatz zu Tiling-Systemen drückt die Form der Dominos aus, dass es egal ist, ob Anlegeschritte zuerst horizontal oder erst vertikal ausgeführt werden.

Beispiel Für die Schachbrettsprache wird wie beim Beispiel für Tiling-Systeme auch hier die quadratische Struktur durch ein separates Symbol a_0 für die Diagonale vorgegeben. Im Gegensatz zu Tiling-Systemen sind für die obere und untere Dreiecksmatrix verschiedene Alphabetsymbole nötig. Dies resultiert daraus, dass bei der Schachbrettsprache das tile, das die obere linke Ecke bestimmt, eindeutig festgelegt ist, aber bei Domino-Systemen alle Zeichen $c \in \Gamma$ mit $\begin{bmatrix} \# & c \\ \# & c \end{bmatrix} \in \Delta$ als obere linke Ecke infrage kommen. Damit nur a_0 in der Ecke oben links bzw. unten rechts erscheinen kann, dürfen alle anderen Dominos nur entweder von der Form $\begin{bmatrix} \# & c \\ c & \# \end{bmatrix}$ oder $\begin{bmatrix} c & \# \\ \# & c \end{bmatrix}$ sein. Genau das wird durch die separaten Zeichen für die obere und untere Dreiecksmatrix garantiert.

ein mögliches picture:

#	#	#	#	#	#	#
#	a_0	b_2	a_2	b_2	a_2	#
#	b_1	a_0	b_2	a_2	b_2	#
#	a_1	b_1	a_0	b_2	a_2	#
#	b_1	a_1	b_1	a_0	b_2	#
#	a_1	b_1	a_1	b_1	a_0	#
#	#	#	#	#	#	#

$$\Theta = \left\{ \begin{array}{cccc} \begin{bmatrix} \# & \# \end{bmatrix}, & \begin{bmatrix} \# & a_0 \end{bmatrix}, & \begin{bmatrix} a_0 & b_2 \end{bmatrix}, & \begin{bmatrix} b_2 & a_2 \end{bmatrix}, \\ \begin{bmatrix} a_2 & b_2 \end{bmatrix}, & \begin{bmatrix} a_2 & \# \end{bmatrix}, & \begin{bmatrix} b_2 & \# \end{bmatrix}, & \begin{bmatrix} b_1 & a_0 \end{bmatrix}, \\ \begin{bmatrix} a_1 & b_1 \end{bmatrix}, & \begin{bmatrix} \# & b_1 \end{bmatrix}, & \begin{bmatrix} \# & a_1 \end{bmatrix}, & \begin{bmatrix} a_0 & \# \end{bmatrix}, \\ \begin{bmatrix} b_1 & a_1 \end{bmatrix}, & \begin{bmatrix} \# \\ \# \end{bmatrix}, & \begin{bmatrix} \# \\ a_0 \end{bmatrix}, & \begin{bmatrix} a_0 \\ b_1 \end{bmatrix}, \\ \begin{bmatrix} b_2 \\ a_0 \end{bmatrix}, & \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}, & \begin{bmatrix} b_2 \\ a_2 \end{bmatrix}, & \begin{bmatrix} \# \\ b_2 \end{bmatrix}, \\ \begin{bmatrix} \# \\ a_2 \end{bmatrix}, & \begin{bmatrix} b_1 \\ a_1 \end{bmatrix}, & \begin{bmatrix} a_1 \\ \# \end{bmatrix}, & \begin{bmatrix} b_1 \\ \# \end{bmatrix}, \\ \begin{bmatrix} a_0 \\ \# \end{bmatrix}, & \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} & & \end{array} \right\}$$

Projektion:

$$\Gamma = \{a_0, a_1, a_2, b_1, b_2\}, \Sigma = \{a, b\}$$

$$\pi : \Gamma \rightarrow \Sigma$$

$$\pi(a_0) = \pi(a_1) = \pi(a_2) = a$$

$$\pi(b_1) = \pi(b_2) = b$$

3 Abschlusseigenschaften

Satz 3.1 Für die Abschlusseigenschaften von $L(TS)$ gilt

- (1.) $L(TS)$ ist unter Projektionen abgeschlossen.
- (2.) $L(TS)$ ist unter $\oplus$ und $\ominus$ abgeschlossen.

(3.) $L(TS)$ ist unter $\oplus^*$ und $\ominus^*$ abgeschlossen.

(4.) $L(TS)$ ist unter $\cup$ und $\cap$ abgeschlossen.

(5.) $L(TS)$ ist **nicht** unter $^-$ abgeschlossen.

Beweis von (5.): Für $L = \{p \in \{a, b\}^{**} \mid s \text{ ist quadratisches picture, } p = s \ominus s\}$ wird gezeigt, dass $L \notin L(TS)$, aber $\bar{L} \in L(TS)$.

$L \notin L(TS)$: Beweis durch Widerspruch. Angenommen, es wäre $L \in L(TS)$. Dann gibt es eine Tiling-System $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$, das L erkennt; insbesondere gibt es eine lokale Sprache $L' = L(\Theta)$ mit $\pi(L') = L$. Betrachte die Klasse aller pictures, bei denen s die Seitenlänge n hat: $L_n = \left\{ p \in \Sigma^{**} \mid p = \begin{bmatrix} s \\ s \end{bmatrix} \right\} 2n$. In s gibt es n^2 viele Positio-

nen, die mit $|\Sigma|$ vielen möglichen Zeichen besetzt sein können. Also gibt es insgesamt $|\Sigma|^{n^2}$ viele pictures in L_n . Die zu L gehörende lokale Sprache L'_n ist von der Form $L'_n = \{p' \in \Gamma^{**} \mid \pi(p') = p, p \in L_n\}$. Es reicht zu überlegen, wie viele Möglichkeiten es gibt, die n -te und $(n+1)$ -te Zeile zu belegen, weil tiles als 2×2 -Blöcke dann ausgetauscht werden können. Mit derselben Überlegung wie oben ergeben sich $|\Gamma|^{2n}$ mögliche Belegungen. Für feste Alphabetgrößen wächst $|\Sigma|^{n^2}$ schneller als $|\Gamma|^{2n}$. Also gibt es ein

n , für das gilt $|\Sigma|^{n^2} > |\Gamma|^{2n}$. Dann gibt es zwei pictures $p = \begin{bmatrix} s_p \\ s_p \end{bmatrix} \in L_n, q = \begin{bmatrix} s_q \\ s_q \end{bmatrix} \in L_n$ mit $s_p \neq s_q$, deren zugehörige pictures in der lokalen Sprache gegeben sind durch

$p' = \begin{bmatrix} s'_p \\ s''_p \end{bmatrix} \in L'_n$ bzw. $q' = \begin{bmatrix} s'_q \\ s''_q \end{bmatrix} \in L'_n$ und die n -te und $(n+1)$ te Zeile von p' und q' seien gleich. Da die untere Zeile von s'_p und s'_q bzw. die obere Zeile von s''_p und s''_q übereinstimmen, können die obere und untere Hälfte des pictures vertauscht werden.

Somit ist $v' = \begin{bmatrix} s'_p \\ s''_q \end{bmatrix} \in L'_n$, aber dann ist auch $\pi(v') = \begin{bmatrix} s_p \\ s_q \end{bmatrix} \in L$. $\nexists$

$\bar{L} \in L(TS)$: Zeige, dass es ein Tiling-System gibt, das $\bar{L}$ erkennt.

$$\bar{L} = \underbrace{\{p \in \Sigma^{**} \mid l_1(p) \neq 2 \cdot l_2(p)\}}_{L_1} \cup \underbrace{\{p \in \Sigma^{**} \mid l_1(p) = 2 \cdot l_2(p), p = \begin{bmatrix} s \\ t \end{bmatrix}, s \neq t\}}_{L_2}$$

$L(TS)$ ist unter Vereinigung abgeschlossen. Also reicht es zu zeigen, dass L_1 und L_2 jeweils durch ein Tiling-System erkennbar sind. L_1 ist ähnlich wie die Schachbrettsprache aufgebaut. Ein Tiling-System, das oben links beginnend einen Schritt nach rechts und zwei Schritte nach unten macht, beschreibt alle pictures der Größe $(n, 2n)$. L_1 umfasst alle pictures, deren Größe nicht $(n, 2n)$ ist, d.h. wo die „Treppenschritte“ nicht in der unteren rechten Ecke enden.

L_2 ist darstellbar als $L_3 \cap (\Sigma^{**} \ominus (L_4 \cap (\Sigma^{**} \oplus L_5 \oplus \Sigma^{**}))) \ominus \Sigma^{**}$ mit

$$L_3 = \{p \in \Sigma^{**} \mid l_1(p) = 2l_2(p)\}$$

$$L_4 = \{p \in \Sigma^{**} \mid l_1(p) = l_2(p) + 1\}$$

$$L_5 = \{p \in \Sigma^{**} \mid l_2(p) = 1 \text{ und } p(1, 1) \neq p(1, l_1(p))\}$$

Da $L(TS)$ unter $\cap, \ominus$ und $\oplus$ abgeschlossen sind, bleibt zu zeigen, dass L_3, L_4 und L_5 durch Tiling-Systeme erkennbar sind. L_3 und L_4 sind ähnlich zu L_1 durch Tiling-Systeme

erkennbar. L_5 ist eine eindimensionale reguläre Sprache, die nach Kapitel 2.5 durch ein Tiling-System erkennbar ist.

4 Äquivalenzaussagen

Satz 4.1 *Es sei L eine zweidimensionale Sprache. Dann sind die folgenden Aussagen äquivalent:*

- (1.) L wird durch eine Tiling-System erkannt.
- (2.) L wird von einem 2OTA erkannt.
- (3.) L wird durch ein Domino-System erkannt.
- (4.) L ist durch eine EMSO-Formel definierbar.
- (5.) L ist darstellbar als komplementfreier regulärer Ausdruck mit Projektion, d.h. $L \in PCFRE$.

Definition Die so definierten Sprachen werden mit *REC*, *recognizable two-dimensional languages* bezeichnet.

Im Folgenden wird gezeigt, dass die Modelle 2OTA, EMSO-Formeln und Domino-Systeme jeweils dieselbe Sprachklasse beschreiben wie Tiling-Systeme.

4.1 $L(2OTA) = L(TS)$

Für die Äquivalenz werden die beiden Inklusionen $L(2OTA) \subseteq L(TS)$ und $L(TS) \subseteq L(2OTA)$ gezeigt.

Beweis $L(2OTA) \subseteq L(TS)$:

Sei $L \subseteq \Sigma^{**}$ und $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ eine 2OTA, der L erkennt. Nun wird ein Tiling-System $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ konstruiert, das L erkennt. Wähle $\Gamma = \Sigma \cup \{\#\} \times Q$. Die lokale Sprache über Γ wird genutzt, um eine Transition eines 2OTAs zu simulieren. Die erste Komponente beschreibt ein picture $p \in L$; in der zweiten Komponente wird der Zustand gespeichert, den $\mathcal{A}$ während eines Laufs einer Position in p zuordnet.

Eine Transition eines 2OTAs hat die Form $\delta(q_u, q_l, p(i, j)) = q$.

	$p(i-1, j)$ q_u
$p(i, j-1)$ q_l	$p(i, j)$ q

In Abhängigkeit von der Position innerhalb eines pictures (Mitte, linker Rand, ...) lassen sich Mengen von tiles definieren, die die Transition von $\mathcal{A}$ simulieren.

$$\begin{aligned}
 \bullet \Theta_m &= \left\{ \begin{array}{|c|c|} \hline (a, r) & (b, s) \\ \hline (c, t) & (d, q) \\ \hline \end{array} \parallel \begin{array}{l} a, b, c, d \neq \#, \\ q \in \delta(s, t, d) \end{array} \right\} \\
 \bullet \Theta_l &= \left\{ \begin{array}{|c|c|} \hline (\#, q_0) & (b, s) \\ \hline (\#, q_0) & (d, q) \\ \hline \end{array} \parallel \begin{array}{l} b, d \neq \#, q_0 \in I, \\ q \in \delta(s, q_0, d) \end{array} \right\}
 \end{aligned}$$

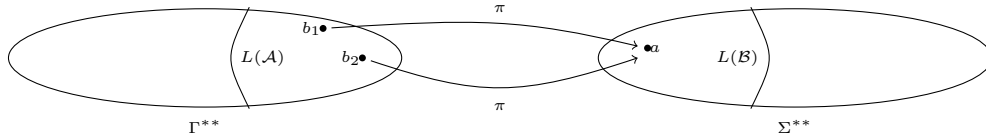
• ...

Die Projektion $\pi : \Sigma \cup \{\#\} \times \Theta \rightarrow \Sigma \cup \{\#\}$ bildet die Einträge eines pictures auf die erste Komponente ab. Hat $\mathcal{A}$ einen akzeptierenden Lauf auf einem picture p , gibt es entsprechende tiles in Θ , die den Lauf kodieren, und durch die Projektion auf die erste Komponente ist $p \in L(\mathcal{T}) \Leftrightarrow p \in L$.

Für den Beweis der Inklusion $L(TS) \subseteq L(2OTA)$ wird folgendes Lemma benötigt.

Lemma 4.2 *$L(2OTA)$ ist unter Projektionen abgeschlossen.*

Beweis Sei $\pi : \Gamma \rightarrow \Sigma$ eine Projektion und $L \subseteq \Gamma^{**}$. Sei weiter $\mathcal{A} = (\Gamma, Q, I, F, \delta)$ ein 2OTA, der L erkennt. Nun wird ein 2OTA $\mathcal{B}$ konstruiert, der $\pi(L)$ erkennt.



Durch die Projektion wird einem Alphabetsymbol $a \in \Sigma$ gegebenenfalls mehrere Alphabetsymbole $b_1, b_2, \dots$ in Γ zugeordnet. $\mathcal{B}$ kann alle Transitionen $\delta(p, q, b_i)$ für ein $b_i \in \Gamma$ von $\mathcal{A}$ simulieren, indem für alle Symbole mit $\pi(b_1) = \pi(b_2) = \dots = a$ die Transition $\delta'(p, q, a)$ in $\mathcal{B}$ eingeführt wird. Der 2OTA $\mathcal{B} = (\Sigma, Q, I, F, \delta')$ mit $\delta'(p, q, a) = \bigcup_{b \in \pi^{-1}(a)} \delta(p, q, b)$ erkennt genau $\pi(L)$.

Nun wird die noch fehlende Inklusion unter Verwendung des oben gezeigten Lemmas bewiesen.

Beweis $L(TS) \subseteq L(2OTA)$:

Sei $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ ein Tiling-System, das $L \subseteq \Sigma^{**}$ erkennt. Da $L(2OTA)$ unter Projektionen abgeschlossen ist, reicht es zu zeigen, dass man einen 2OTA $\mathcal{A}$ konstruieren kann, der die lokale Sprache $L(\mathcal{T})$ erkennt.

Der 2OTA $\mathcal{A} = (\Gamma, Q, I, F, \delta)$ mit

- $Q = \Theta$
- $I = \left\{ \begin{bmatrix} \# & b \\ c & d \end{bmatrix} \mid b, c, d \in \Gamma \cup \{\#\} \text{ und } \begin{bmatrix} \# & b \\ c & d \end{bmatrix} \in \Theta \right\}$
- $F = \left\{ \begin{bmatrix} a & \# \\ \# & \# \end{bmatrix} \parallel \begin{bmatrix} a & \# \\ \# & \# \end{bmatrix} \in \Theta \right\}$
- $\delta : Q \times Q \times \Sigma \rightarrow 2^Q$ so dass für alle $\begin{bmatrix} x & y \\ a & b \end{bmatrix}, \begin{bmatrix} z & a \\ t & c \end{bmatrix} \in Q$:

$$\delta \left(\begin{bmatrix} x & y \\ a & b \end{bmatrix}, \begin{bmatrix} z & a \\ t & c \end{bmatrix}, a \right) = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \parallel \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in Q \right\}$$

simuliert Tiling-Schritte. Ein picture ist genau dann in $L(\Theta)$ enthalten, wenn durch Anlegeschritte von tiles in Θ ein rechteckiges picture konstruiert werden kann, dessen Rand aus $\#$ besteht und dessen innere Positionen mit Zeichen aus Γ besetzt sind. Das entspricht einem akzeptierenden Lauf von $\mathcal{A}$, d.h. die untere rechte Ecke ist ein Endzustand; oberer und linker Rand sind Startzustände. Eine Transition ist möglich, wenn sich zwei tiles geeignet überlappen:

	x	y
z	a	b
t	c	d

4.2 $L(DS) = L(TS)$

Tiling-Systeme und Domino-Systeme sind ähnlich aufgebaut, so dass für die beiden Inklusionen $L(DS) \subseteq L(TS)$ und $L(TS) \subseteq L(DS)$ die lokale Sprache in eine hv-lokale Sprache bzw. umgekehrt überführt wird.

Beweis $L(DS) \subseteq L(TS)$:

Sei $\mathcal{D} = (\Sigma, \Gamma, \Delta, \pi)$ ein Domino-System mit $L' = L(\Delta)$. Konstruiere ein Tiling-System $\mathcal{T} = (\Sigma, \Gamma_{\mathcal{T}}, \Theta, \pi_{\mathcal{T}})$ mit $L(\Theta) = L'$.

Ein tile ist eindeutig durch 4 Dominos festgelegt, nämlich zwei horizontale und zwei vertikale Dominos, die sich so überlappen, dass ein Block der Größe $(2, 2)$ konstruiert wird. Θ umfasst alle tiles, die durch Anlegeschritte von 4 Dominos konstruiert werden können.

$$\Theta = \left\{ \theta \in (\Gamma \cup \{\#\})^{2 \times 2} \mid \theta \neq \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & \# \\ \hline \end{array}, (B_{1,2}(\theta) \cup B_{2,1}(\theta)) \subseteq \Delta \right\}$$

Hieraus folgt direkt, dass $L(\Theta) = L'$ und damit auch $L(\mathcal{T}) = L(\mathcal{D})$ gilt.

Beweis $L(TS) \subseteq L(DS)$:

Dominos weisen weniger Struktur auf als tiles. Da hv-lokale Sprachen eine echte Teilmenge von lokalen Sprachen bilden, können Domino-Systeme Tiling-Systeme nur durch eine zusätzliche Projektion π' simulieren. Die Idee des Beweises besteht darin, alle möglichen Tiling-Schritte durch Dominos zu kodieren und mit einer Projektion auf die lokale Sprache des Tiling-Systems abzubilden. Sei also $\mathcal{T} = (\Sigma', \Sigma, \Theta, \pi)$ ein Tiling-System und $\mathcal{D} = (\Sigma', \Gamma, \Delta, \pi)$ das zu konstruierende Domino-System.

Tiling-Schritte sind möglich, wenn die benachbarten Einträge zweier tiles übereinstimmen:

$$\begin{array}{ccc} \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array} & \text{und} & \begin{array}{|c|c|} \hline a_2 & a_5 \\ \hline a_4 & a_6 \\ \hline \end{array} & \rightsquigarrow & \begin{array}{|c|c|c|} \hline a_1 & a_2 & a_5 \\ \hline a_3 & a_4 & a_6 \\ \hline \end{array} \\ \\ \text{bzw.} & & \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array} & \text{und} & \begin{array}{|c|c|} \hline a_3 & a_4 \\ \hline a_5 & a_6 \\ \hline \end{array} & \rightsquigarrow & \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline a_5 & a_6 \\ \hline \end{array} \end{array}$$

Alle möglichen Tiling-Schritte werden simuliert, indem alle möglichen Anlegeschritte über einem größeren Alphabet Γ nachgebildet werden. Für die obigen beiden Tiling-

Schritte werden also die Dominos $\begin{array}{|c|c|c|c|} \hline a_1 & a_2 & a_2 & a_5 \\ \hline a_3 & a_4 & a_4 & a_6 \\ \hline \end{array}$ bzw. $\begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline a_3 & a_4 \\ \hline a_5 & a_6 \\ \hline \end{array}$ in Δ aufnehmen.

Das Domino-System sieht also wie folgt aus:

- $\Gamma := \Theta$
- $\Delta = \left\{ \begin{array}{|c|c|c|c|} \hline a_1 & a_2 & b_1 & b_2 \\ \hline a_3 & a_4 & b_3 & b_4 \\ \hline \end{array}, \begin{array}{|c|c|} \hline c_1 & c_2 \\ \hline c_3 & c_4 \\ \hline d_1 & d_2 \\ \hline d_3 & d_4 \\ \hline \end{array} \mid \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array}, \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array}, \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array}, \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array} \in \Gamma \right\}$
 $\cup \left\{ \begin{array}{|c|c|c|c|} \hline b_2 & \# & \# & \# \\ \hline b_4 & \# & \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline d_3 & d_4 \\ \hline \# & \# \\ \hline \# & \# \\ \hline \# & \# \\ \hline \end{array} \mid \begin{array}{|c|c|} \hline b_2 & \# \\ \hline b_4 & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline d_3 & d_4 \\ \hline \# & \# \\ \hline \end{array} \in \Gamma \right\}$
- $\pi' : \Gamma \rightarrow \Sigma$
 $\begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array} \mapsto a_1$

Die Wahl der Dominos hat zur Folge, dass bei Anlegeschritten alle Einträge bis auf den oberen linken eines tiles dupliziert werden. Wendet man die Konstruktion auf ein picture an, werden also alle Einträge dupliziert bis auf die rechte Spalte und untere Zeile eines pictures. Um auch diese Einträge zu verdoppeln, werden für diese Einträge spezielle Dominos zu Δ hinzugefügt, bei denen eine Hälfte des Dominos nur mit $\#$ gefüllt sind. Mittels einer zusätzlichen Projektion auf den oberen linken Eintrag einer jeden Hälfte eines Dominos wird die Duplizierung der Einträge rückgängig gemacht.

Beispiel:

$$\hat{p} = \begin{array}{|c|c|c|c|} \hline \# & \# & \# & \# \\ \hline \# & p_1 & p_2 & \# \\ \hline \# & \# & \# & \# \\ \hline \end{array} \rightsquigarrow \hat{p}' = \begin{array}{|c|c|c|c|c|c|} \hline \# & \# & \# & \# & \# & \# \\ \hline \# & p_1 & p_1 & p_2 & p_2 & \# \\ \hline \# & p_1 & p_1 & p_2 & p_2 & \# \\ \hline \# & \# & \# & \# & \# & \# \\ \hline \# & \# & \# & \# & \# & \# \\ \hline \# & \# & \# & \# & \# & \# \\ \hline \end{array}$$

Wendet man die Projektion auf $\hat{p}'$ an, erhält man genau $\hat{p}$ zurück.

Korrektheitsbeweis:

zu zeigen: $\pi(L(\Delta)) = L(\Theta)$

„ $\supseteq$ “: Sei $p \in L(\Theta)$, $p \in \Gamma^{m \times n}$. Zeige $p' \in L(\Delta)$. p' erhält man aus p durch Anwenden der Konstruktionsvorschrift für Dominos (vgl. obiges Beispiel). Somit gilt für die Einträge von $p' \in L(\Delta)$ folgendes:

- $p'(i, j) = \begin{array}{|c|c|} \hline p(i, j) & p(i, j+1) \\ \hline p(i, j+1) & p(i+1, j+1) \\ \hline \end{array} \quad 1 \leq i < m, 1 \leq j < n$
- $p'(i, n) = \begin{array}{|c|c|} \hline p(i, n) & \# \\ \hline p(i, n) & \# \\ \hline \end{array} \quad 1 \leq i < m$
- $p'(m, j) = \begin{array}{|c|c|} \hline p(m, j) & p(m, j+1) \\ \hline \# & \# \\ \hline \end{array} \quad 1 \leq j < n$

$$\bullet p'(m, n) = \begin{array}{|cc|} \hline p(m, n) & \# \\ \hline \# & \# \\ \hline \end{array}$$

Mit der Konstruktion ist klar, dass $p' \in L(\Delta)$ gilt, denn die Einträge benachbarter Felder sind gleich. Weiter ist $\pi(p') = p$, da jeder Eintrag $p(i, j)$ in p in einem Eintrag $p(i, j)$ in $p'(i, j)$ resultiert, nämlich den oben links.

„ $\subseteq$ “: Sei $p' \in L(\Delta)$. Zeige $\pi(p') \in L(\Theta)$. Sei $q \in B_{2,2}(\hat{p}')$. Da benachbarte Einträge nach Definition der Dominos gleich sind, hat q folgende Form:

a_1	b_1	b_1	b_2
c_1	d_1	d_1	d_2
c_1	d_1	d_1	d_2
c_3	d_3	d_3	d_4

$$\text{wobei } \begin{array}{|cc|} \hline a_1 & b_1 \\ \hline c_1 & d_1 \\ \hline \end{array}, \begin{array}{|cc|} \hline c_1 & d_1 \\ \hline c_3 & d_3 \\ \hline \end{array}, \begin{array}{|cc|} \hline b_1 & b_2 \\ \hline d_1 & d_2 \\ \hline \end{array}, \begin{array}{|cc|} \hline d_1 & d_2 \\ \hline d_3 & d_4 \\ \hline \end{array} \in \Theta.$$

$$\text{Dann ist } \pi(q) = \begin{array}{|cc|} \hline a_1 & b_1 \\ \hline c_1 & d_1 \\ \hline \end{array} \in \Theta.$$

4.3 $L(EMSO) = L(TS)$

Mit den Inklusionen $L(EMSO) \subseteq L(TS)$ und $L(TS) \subseteq L(EMSO)$ folgt die Gleichheit.

Beweis $L(TS) \subseteq L(EMSO)$:

Definiere zu einem gegebenen Tiling-System $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ einen EMSO-Satz φ mit $L(\mathcal{T}) = L(\varphi)$. O.B.d.A. kann angenommen werden, dass $\mathcal{T}$ von folgender Form ist:

- $\Gamma = \Sigma \times Q$
- $\pi : \Sigma \times Q \rightarrow \Sigma$

Dabei ist $p' \in L(\Theta)$ von der Form $p' = p \times c$ mit p und c als pictures derselben Größe. Die lokale Sprache wird in der Komponente Q kodiert. Somit genügt es zu zeigen, dass die lokale Sprache $L(\Theta)$ durch einen Satz dargestellt werden kann.

Die lokale Sprache wird mit Mengenvariablen $X_1, \dots, X_k$ kodiert. Für $Q = \{q_1, \dots, q_k\}$ bedeutet $X_l(x)$, dass $c(x) = q_l$. Weiter gilt

$$\begin{aligned} p \in L &\Leftrightarrow \text{gibt ein passendes picture } c \text{ mit } p' = p \times c, p' \in L(\Theta) \\ &\Leftrightarrow \text{gibt ein passendes picture } c \text{ mit } p' = p \times c, B_{2,2}(\hat{p}') \subseteq \Theta \\ &\Leftrightarrow \exists X_1 \dots \exists X_k (\varphi_{\text{Partition}}(X_1, \dots, X_k)) \text{ und das picture } c \text{ wird kodiert als} \\ &\quad c(i, j) = q_l \Leftrightarrow (i, j) \in X_l \text{ und für } p' = p \times c, B_{2,2}(\hat{p}') \subseteq \Theta \end{aligned}$$

Dabei drückt $\varphi_{\text{Partition}}(X_1, \dots, X_k) = \forall z ((\bigvee_{i=1}^k X_i(z)) \wedge \bigwedge_{\substack{1 \leq i, j \leq k, \\ i \neq j}} \neg(X_i(z) \wedge X_j(z)))$ aus,

dass an jeder Position eines pictures genau ein Eintrag mit einem Zeichen aus Q steht.

Ein tile lässt sich mit einer Formel beschreiben, die zum einem das Format eines 2×2 Blocks, und zum anderen den Inhalt definiert.

Das Format hängt von der Position des tiles in einem picture $\hat{p}$ ab; in Abhängigkeit der Randbegrenzung lassen sich folgende Klassen von Blöcken (t: oben, l: links, r: rechts, b: unten, tl: oben links, usw.) und Formeln definieren:

$$\Theta = \Theta_m \cup \Theta_t \cup \Theta_b \cup \Theta_l \cup \Theta_r \cup \Theta_{tl} \cup \Theta_{tr} \cup \Theta_{bl} \cup \Theta_{br}$$

x_1	x_2	$\psi_m(x_1, x_2, x_3, x_4) = x_1 S_1 x_3 \wedge x_1 S_2 x_2 \wedge x_3 S_2 x_4 \wedge x_2 S_1 x_4$
x_3	x_4	

$\#$	$\#$	$\psi_t(x_3, x_4) = x_3 S_2 x_4 \wedge \neg \exists x (x S_1 x_3) \wedge \neg \exists x (x S_1 x_4)$
x_3	x_4	

$\vdots$

Der Inhalt eines 2×2 Blocks $\begin{smallmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{smallmatrix}$ besteht aus Einträgen der Form (a, q_i) . In der ersten Komponente wird das „tatsächliche“ und in der zweiten Komponente das lokale picture gespeichert. Ein Eintrag ist damit darstellbar als $\varphi_{\theta_i}(x) = P_a(x) \wedge X_l(x)$ für $i \in \{1, 2, 3, 4\}$. Jetzt lassen sich in Abhängigkeit des Formats tiles¹ definieren:

$$\begin{aligned} \chi_m &= \underbrace{\psi_m(x_1, x_2, x_3, x_4)}_{\text{Format: } \begin{smallmatrix} x_1 & x_2 \\ x_3 & x_4 \end{smallmatrix}} \rightarrow \bigvee_{\begin{smallmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{smallmatrix} \in \Theta_m} \bigwedge_{i=1}^4 \varphi_{\theta_i}(x_i) \\ \chi_t &= \underbrace{\psi_t(x_3, x_4)}_{\text{Format: } \begin{smallmatrix} \# & \# \\ x_3 & x_4 \end{smallmatrix}} \rightarrow \bigvee_{\begin{smallmatrix} \# & \# \\ \theta_3 & \theta_4 \end{smallmatrix} \in \Theta_t} (\varphi_{\theta_3}(x_3) \wedge \varphi_{\theta_4}(x_4)) \\ &\vdots \end{aligned}$$

Mit den eben definierten Formeln lässt sich ein EMSO-Satz φ angeben, der $L(\mathcal{T}) = L(\varphi)$ erfüllt.

$$\varphi = \exists X_1 \dots \exists X_k (\varphi_{\text{Partition}} \wedge \forall x_1 \dots \forall x_4 (\chi_m \wedge \chi_t \wedge \chi_b \wedge \chi_l \wedge \chi_r \wedge \chi_{tl} \wedge \chi_{tr} \wedge \chi_{br} \wedge \chi_{bl}))$$

Beweis $L(\text{EMSO}) \subseteq L(\text{TS})$:

Für den Beweis werden weitere Definitionen benötigt, die ausgelassen wurden. Die Inklusion wird daher hier nicht gezeigt.

5 Unentscheidbarkeit

Satz 5.1 *Das Leerheitsproblem für REC ist unentscheidbar.*

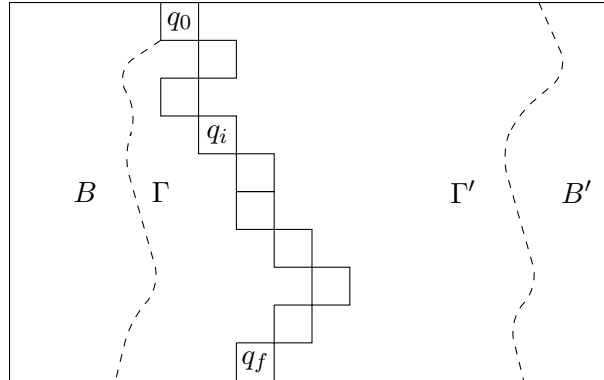
¹Die freien Variablen der Formeln χ_i sind zwecks Übersichtlichkeit weggelassen und entsprechen denen von ψ_i .

Beweis Ein Tiling-System kann eine Turingmaschine $\mathcal{M}$ simulieren. Es wird eine Menge von tiles $\Theta_{\mathcal{M}}$ derart konstruiert, dass $L(\mathcal{M}) = \emptyset \Leftrightarrow L(\Theta_{\mathcal{M}}) = \emptyset$. Da $\text{LOC} \subset \text{REC}$ (vgl. Schachbrettsprache) und da das Leerheitsproblem für Turingmaschinen unentscheidbar ist, folgt, dass das Leerheitsproblem für LOC und damit auch für die Sprachklasse REC unentscheidbar ist.

Sei $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \delta, F)$ eine Turingmaschine. Ein akzeptierender Lauf von $\mathcal{M}$ auf einer Eingabe v_0 ist eine Folge von Konfigurationen mit $q_f \in F$.

$$q_0 v_0 \vdash \dots \vdash u_i q v_i \vdash \dots \vdash u_f q_f v_f$$

Der Lauf lässt sich durch eine Menge von tiles simulieren. Eine einzelne Konfiguration von $\mathcal{M}$, d.h. ein String $u_i q v_i$, stellt in einem picture eine Zeile dar. Die einzelnen Konfigurationen werden links und rechts so mit Blank-Symbolen ergänzt, dass sich ein rechteckiges picture ergibt. Die i -te Konfiguration $u_i q v_i$ hat also das Format $w_i = \sqcup \dots \sqcup u_i q v_i \sqcup \dots \sqcup$. Insgesamt ergibt sich für einen Lauf folgendes picture:



Ein Rechenschritt $u_i q v_i \vdash u_{i+1} p v_{i+1}$ von $\mathcal{M}$ wird dargestellt als $w_i = B^* \underbrace{u_i}_{\in \Gamma^*} q \underbrace{v_i}_{\in \Gamma'^*} B^*$ und $w_{i+1} = B^* \underbrace{u_{i+1} p}_{\in \Gamma'^*} \underbrace{v_{i+1}}_{\in \Gamma'^*} B^*$. Dabei steht q an Position j der i -ten Zeile und p an der Position $j-1, j$ oder $j+1$ der $(i+1)$ -ten Zeile je nachdem, ob sich der Kopf von $\mathcal{M}$ nach links, gar nicht oder nach rechts bewegt. Ein picture p_w entspricht also einem akzeptierenden Lauf und p_w definiert eine Menge von tiles, die diesen Lauf kodieren. Die Menge aller akzeptierenden Läufe von $\mathcal{M}$ lässt sich durch eine Menge $\Theta_{\mathcal{M}}$ von tiles kodieren.

Korrektheit:

Es ist $(L(\mathcal{M}) = \emptyset \Leftrightarrow L(\Theta_{\mathcal{M}}) = \emptyset) \Leftrightarrow (L(\mathcal{M}) \neq \emptyset \Leftrightarrow L(\Theta_{\mathcal{M}}) \neq \emptyset)$. Für den Korrektheits-

beweis wird die zweite Äquivalenzaussage genutzt:

$$\begin{aligned}
\text{„} \Rightarrow \text{“} \quad L(\mathcal{M}) \neq \emptyset &\Rightarrow \text{es gibt einen akzeptierenden Lauf von } \mathcal{M} \text{ auf einer Eingabe } w \\
&\Rightarrow \text{es ex. picture } p_w \text{ derart, dass } B_{2,2}(\hat{p}_w) \text{ kompatibel mit den} \\
&\quad \text{Transitionen von } \mathcal{M} \text{ ist} \\
&\Rightarrow B_{2,2}(\hat{p}_w) \subseteq \Theta_{\mathcal{M}} \\
&\Rightarrow L(\Theta_{\mathcal{M}}) \neq \emptyset \\
\text{„} \Leftarrow \text{“} \quad L(\Theta_{\mathcal{M}}) \neq \emptyset &\Rightarrow \text{aus den tiles in } \Theta \text{ lässt sich ein picture konstruieren, das einen Lauf} \\
&\quad \text{einer Turingmaschine } \mathcal{M} \text{ auf einer Eingabe } w \text{ kodiert} \\
&\Rightarrow \mathcal{M} \text{ akzeptiert } w \\
&\Rightarrow L(\mathcal{M}) \neq \emptyset
\end{aligned}$$

6 Fazit

Die Sprachklasse REC wurde eingeführt und stellt eine sinnvolle Erweiterung von ein- auf zweidimensionale Sprachen dar. REC weist starke Abschlusseigenschaften auf und es gibt, wie im eindimensionalen Fall, viele Modelle, die diese Sprachklasse beschreiben. Ungünstig ist es, dass REC nicht unter Komplement abgeschlossen ist. Desweiteren sind die Modelle nichtdeterministisch. Wenn man die Modelle auf eine deterministische Variante reduziert, wird die erkannte Sprachklasse deutlich kleiner.

Literatur

- [GR97] D. Giammarresi, A. Restivo: Two-dimensional Languages. In Handbook of Formal Languages, G. Rosenberg and A. Salomaa (Eds), Vol. III, Seite 215-268. Springer Verlag, 1997.
- [PO04] T. Polcar, B. Melichar: A Two-dimensional Online Tessellation Automata Approach to Two-dimensional Pattern Matching. Proceedings of the Eindhoven FASTAR Days 2004.