On some Recognizable Picture-languages *

Klaus Reinhardt

Wilhelm-Schickhard Institut für Informatik, Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
e-mail: reinhard@informatik.uni-tuebingen.de

Abstract. We show that the language of pictures over $\{a, b\}$, where all occurring b's are connected is recognizable, which solves an open problem in [Mat98]. We generalize the used construction to show that monocausal deterministically recognizable picture languages are recognizable, which is surprisingly nontrivial. Furthermore we show that the language of pictures over $\{a, b\}$, where the number of a's is equal to the number of b's is nonuniformly recognizable.

1 Introduction

In [GRST94] pictures are defined as two-dimensional rectangular arrays of symbols of a given alphabet. A set (language) of pictures is called recognizable if it is recognized by a finite tiling system. It was shown in [GRST94] that a picture language is recognizable iff it is definable in existential monadic second-order logic. In [Wil97] it was shown that star-free picture expressions are strictly weaker than first-order logic. A comparison to other regular and context-free formalisms to describe picture languages can be found in [Mat97,Mat98]. Characterizations of the recognizable picture languages by automata can be found in [IN77] and [GR96], where also the subclasses, which are defined by a restriction from nondeterminism to determinism or unambiguity are considered.

We show in chapter 2 that the language of pictures over $\{a, b\}$, where all occurring b's are connected is recognizable, which solves an open problem in [Mat98]. (Connectedness is not recognizable in general [FSV95].) The technique which is used here is generalized in the following chapter to show that monocausal deterministically recognizable languages are recognizable. The notion of deterministic recognizability, which we use is stronger than the determinism in [GR96], has more closure properties (for example rotation), which promises practical relevance.

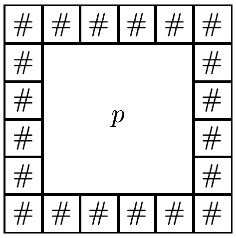
Furthermore we show in the last chapter that the language of pictures over $\{a, b\}$, where the number of a's is equal to the number of b's is nonuniformly recognizable. Hereby we use counters similar to those used in [Für82].

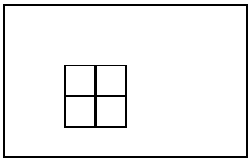
Definition 1. *[GRST94] A* picture *over Σ is a two-dimensional array of elements of Σ. The set of pictures of size (m, n) is denoted by $\Sigma^{m,n}$. A* picture

* This research has been supported by the DFG Project La 618/3-1 KOMET.

language *is a subset of* $\Sigma^{*,*} := \bigcup_{m,n\geq 0} \Sigma^{m,n}$. *For a* $p \in \Sigma^{m,n}$*, we have* $\hat{p} \in \Sigma^{m+2,n+2}$ *adding a frame of symbols* $\# \notin \Sigma$.

Let $T_{2,2}(p)$ *be the set of all sub-pictures of* p *with size* $(2,2)$.

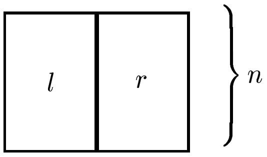
$\hat{p} :=$ 



A picture language $L \subseteq \Gamma^{*,*}$ *is called* local *if there is a* Δ *with* $L = \{p \in \Gamma^{*,*} | T_{2,2}(\hat{p}) \subset \Delta\}$. *A picture language* $L \subseteq \Sigma^{*,*}$ *is called* recognizable *if there is a mapping* $\pi : \Gamma \to \Sigma$ *and a local language* $L' \subset \Gamma^{*,*}$ *with* $L = \pi(L')$.

A necessary condition for recognizability is the following:

Lemma 1. *[Mat98] Let* $L \subseteq \Gamma^{*,*}$ *be recognizable and* $(M_n \subseteq \Gamma^{n,*} \times \Gamma^{n,*})$ *be pairs with* $\forall n, \forall (l,r) \in M_n$ $lr \in L$ *and* $\forall (l,r) \neq (l',r') \in M_n$ $lr' \notin L$ *or* $l'r \notin L$, *then* $|M_n| \in 2^{O(n)}$.

$lr =$ 

Considering pictures where the width is in $2^{\omega(n)}$ for the height n, we can find $2^{\omega(n)}$ pairs (l,r) such that the number of a's in lr is equal to the number of b's in lr but all the l's have a different differences of numbers of a's and b's. By contradiction we get the following:

Corollary 1. *The language of pictures over* $\{a,b\}$*, where the number of a's is equal to the number of b's (and where the size* (n,m) *is only restricted to* $f^{-1}(n) \leq m \leq f(n)$ *for a function* $f \in 2^{\omega(n)}$*) is not recognizable.*

In Section 4 we will see that we can not make such a conclusion, if we restrict the width of the pictures to be at most exponential to the height (and vice versa), by showing nonuniform recognizability in this case.

2 Connectedness in a Planar Grid

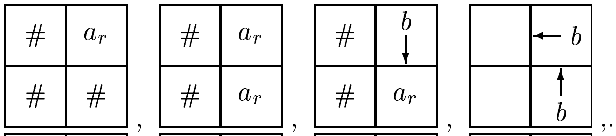
An interesting question for picture recognition is, whether an object is 'in one piece', which means the subgraph of the grid having a special color respectively letter is connected.

Theorem 1. *The language of pictures over* $\{a,b\}$*, where all occurring b's are connected is recognizable.*

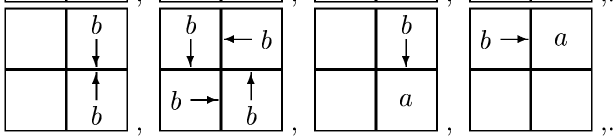
Proof. Since the recognizable languages are closed under concatenation according to [GRST94], it suffices to show that the language of pictures p_r over $\{a,b\}$, where all occurring b's are connected and one of them touches the left side is recognizable. Then we can concatenate with the language of pictures p_l consisting only of a's. The result is the language of pictures $p_l p_r$ of the theorem.

The idea is that the b's are connected if and only if they can be organized as a tree being rooted at the lowest b on the left side. (All a's under this b are distinguished from the other a's.) Hereby π and Δ are constructed in a way such that every g in Γ with $\pi(g) = b$ encodes the direction to the parent node.

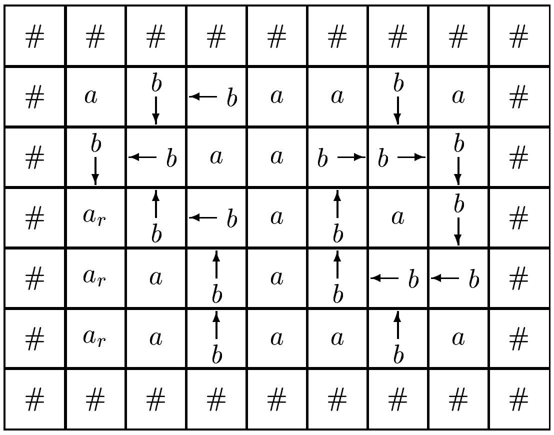
To achieve this Δ has to contain tiles having for example the form



but contain **no** tiles of the form

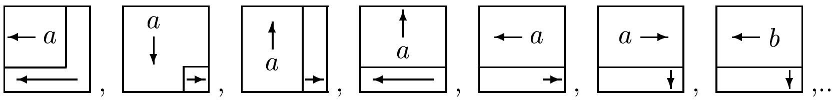


which would build a cycle or not connect to parent nodes. But the picture to the right side shows that it is not so easy: A problem of this naive approach is that cycles could exist independently from the root.



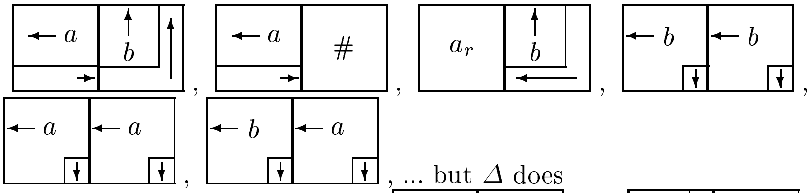
To solve this problem we additionally encode *tentacles* into each cell, where they can occur at the lower and the right side and must occur at the lower right corner. (This means we interpret one cell as a two-dimensional structure of four cells like embedding a grid into a grid with double resolution.) These tentacles also have to build trees, which can have their roots at any # or a_r. Furthermore we do not allow a tentacle crossing a connection of the tree of b's. Each lower right side of a cell must be a tentacle part, which needs a way to a # or a_r, therefore the b's can not have a cycle around such a spot and thus no cycle at all. Analogously we have to avoid cycles in a tentacle tree. Therefore we also organize the a's in trees which are rooted to the tree of b's. This means the tree of b's and a's and the tentacle trees completely intrude the spaces between the other tree and hereby avoid any cycle.

The alphabet Γ contains for example

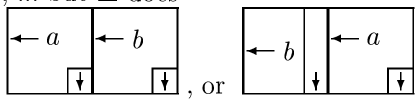


The first kind allows 2 possible parent directions for the a (the other 2 direction would cross the tentacle) and 4 possible parent directions for the tentacle, which lead to 8 possible combinations; the second kind allows 4 possible parent directions for the a and 2 possible parent directions for the tentacle, which again leads to 8 possible combinations; the third and fourth kind allow 3 possible parent directions for the a and 3 possible parent directions for the tentacle, which leads to 9 possible combinations. This means 34 elements for a and the same number for b thus together with a_r we have $|\Gamma| = 69$.

Our tiling Δ allows neighboring cells if tentacles have a parent direction pointing to a tentacle, # or a_r, if furthermore b's have a parent direction pointing to a b or downward to an a_r and a's have a parent direction pointing to an a or a b like for example (only half of a tile is shown):

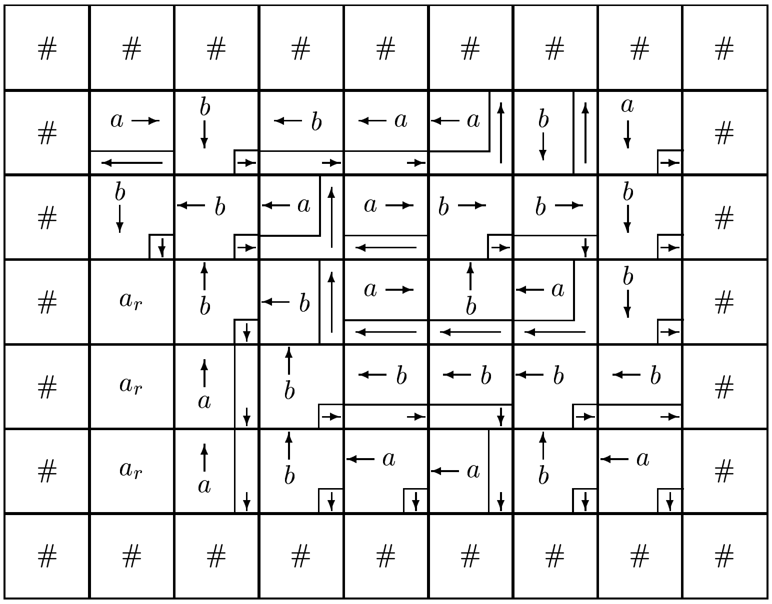


, ... but Δ does not allow tiles containing for example



, or .

The picture $\hat{p}$ could for example look like:

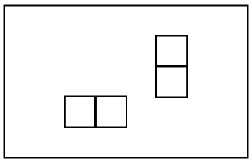


3 The Recognition of a Picture as a Deterministic Process

Recognizing a given picture p can be viewed as a process finding a picture p' over Γ in the local language with $\pi(p') = p$. One major feature for recognizable languages is that this process is nondeterministic.

For practical applications however, we would like to have an appropriate deterministic process starting with a given picture p over Σ and ending with the local p' over Γ. The intermediate configurations are pictures over $\Sigma \cup \Gamma$. One step is a replacement of an $s \in \Sigma$ by a $g \in \Gamma$ with $s = \pi(g)$, which can be performed only if it is locally the only possible choice. This means formally:

Definition 2. *Let $\Sigma \cap \Gamma = \emptyset$, $\pi : \Gamma \to \Sigma$ and $\Delta \subseteq (\Gamma \cup \{\#\})^{1,2} \cup (\Gamma \cup \{\#\})^{2,1}$, which means we consider two kinds of tiles (We conjecture that using 2×2-tiles would make non recognizable picture languages deterministically recognizable):*



Extend Δ to $\Delta' = \Delta \cup \{ \begin{array}{|c|}\hline s\\\hline r\\\hline\end{array}, \begin{array}{|c|}\hline s\\\hline f\\\hline\end{array}, \begin{array}{|c|}\hline g\\\hline r\\\hline\end{array}, \begin{array}{|c|c|}\hline q & o\\\hline\end{array}, \begin{array}{|c|c|}\hline q & d\\\hline\end{array}, \begin{array}{|c|c|}\hline e & o\\\hline\end{array} \mid \begin{array}{|c|}\hline g\\\hline f\\\hline\end{array}, \begin{array}{|c|c|}\hline e & d\\\hline\end{array} \in \Delta, s = \pi(g), r = \pi(f), q = \pi(e), o = \pi(d), \}$ *by also allowing the image symbols in the tiling.*

For two intermediate configurations $p, p' \in (\Sigma \cup \Gamma \cup \{\#\})^{m,n}$, $n, m > 0$ we allow a replacement step $p \underset{(\Delta,\pi)}{\Longrightarrow} p'$ if for all $i \leq m, j \leq n$ we have either $p(i,j) = p'(i,j)$ or $p(i,j) = \pi(p'(i,j))$ and all of the 4 tiles containing this $p'(i,j)$ namely $\begin{array}{|c|}\hline p(i,j\text{-}1)\\\hline p'(i,j)\\\hline\end{array}, \begin{array}{|c|}\hline p'(i,j)\\\hline p(i,j{+}1)\\\hline\end{array}, \begin{array}{|c|c|}\hline p(i\text{-}1,j) & p'(i,j)\\\hline\end{array}$ *and* $\begin{array}{|c|c|}\hline p'(i,j) & p(i{+}1,j)\\\hline\end{array}$ *are in Δ' and if the choice of $p'(i,j)$ was 'forced', that means there is no other $g \neq p'(i,j)$ in Γ with $p(i,j) = \pi(g)$ such that replacing $p(i,j)$ in p by g would result in each of the 4 tiles containing this g is in Δ'. If the choice of $p'(i,j)$ was forced even if 3 of the neighbors where in Σ (or regarded as their image of π), then the replacement step $p \underset{m(\Delta,\pi)}{\Longrightarrow} p'$ is called* monocausal.

The accepted language is $\mathcal{L}_d(\Delta, \pi) := \{p \in \Sigma^{*,*} | \hat{p} \overset{*}{\underset{(\Delta,\pi)}{\Longrightarrow}} p' \in (\Gamma \cup \{\#\})^{*,*}\}$.

and analogously $\mathcal{L}_{md}(\Delta, \pi) := \{p \in \Sigma^{*,*} | \hat{p} \overset{*}{\underset{m(\Delta,\pi)}{\Longrightarrow}} p' \in (\Gamma \cup \{\#\})^{*,*}\}$. *A picture language $L \subseteq \Sigma^{*,*}$ is called* deterministically recognizable *if there are Δ, π with $L = \mathcal{L}_d(\Delta, \pi)$ and analogously* monocausal deterministically recognizable *if $L = \mathcal{L}_{md}(\Delta, \pi)$.*

Clearly $\mathcal{L}_{md}(\Delta, \pi) \subseteq \mathcal{L}_d(\Delta, \pi) \subseteq \mathcal{L}(\Delta, \pi)$. Furthermore it is easy to see that $\underset{(\Delta,\pi)}{\Longrightarrow}$ is confluent on pictures (and their intermediate configurations), which are in $\mathcal{L}_d(\Delta, \pi)$, if we regard possible replacements as voluntarily. (But even if the generated picture $p' \in \Gamma$ is unambiguous, this does not mean that that a deterministically recognizable language is unambiguously recognizable in the sense of [GR96], since the simulation of order of replacements might be ambiguous.) This gives us a simple algorithm to simulate the process by adding those neighbors of a cell, which has just been replaced, to a queue if they are now forced to be replaced and not already in the queue.

Corollary 2. *Deterministically recognizable picture languages can be accepted in linear time.*

As an exercise for the following Theorem 2 we show:

Lemma 2. *The language of pictures over $\{a, b\}$, where all occurring b's are connected to the bottom line is monocausal deterministically recognizable.*

Proof. The language is $\mathcal{L}_d(\Delta, \pi)$ for $\pi(x_i) = x$ and $\Delta = \{$
$\begin{array}{|c|}\hline b_c\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_i & b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_i & \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_c & b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_i & a_c\\\hline\end{array}$, $\begin{array}{|c|}\hline \#\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|}\hline b_i\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|}\hline a_c\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|}\hline b_i\\\hline a_c\\\hline\end{array}$,
$\begin{array}{|c|}\hline a_c\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|}\hline \#\\\hline a_c\\\hline\end{array}$, $\begin{array}{|c|}\hline a_c\\\hline a_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_c & \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & a_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_c & a_c\\\hline\end{array}$, $|i \in \{c, u\}\}$. Clearly a's can only be replaced by a_c. The b's could possibly be b_c or b_u. In the first step only the b's at the bottom line can be replaced by b_c since b_u can not occur there. Then in the following steps b's, which are neighbors of an b_c can be replaced by b_c since a b_u can not occur beside a b_c. In this way all connected b's are replaced by b_c.

Theorem 2. *The language of pictures over $\{a, b\}$, where all occurring b's are connected is (monocausal) deterministically recognizable.*

Proof. The language is $\mathcal{L}_d(\Delta, \pi)$ for $\pi(x_i) = x$ and $\Delta = \{$
$\begin{array}{|c|}\hline a_{sr}\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & a_{sr}\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_{sr} & a_{sr}\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_{sr} & \#\\\hline\end{array}$, $\begin{array}{|c|}\hline a_{sl}\\\hline a_{sr}\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_{sl} & \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_{sl} & a_{sl}\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & a_{sl}\\\hline\end{array}$, $\begin{array}{|c|}\hline a_{sr}\\\hline a_{sl}\\\hline\end{array}$, $\begin{array}{|c|}\hline \#\\\hline a_{sl}\\\hline\end{array}$, $\begin{array}{|c|}\hline \#\\\hline a_{sr}\\\hline\end{array}$,
$\begin{array}{|c|}\hline b_c\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|}\hline b_c\\\hline a_{sr}\\\hline\end{array}$, $\begin{array}{|c|}\hline b_c\\\hline a_{sl}\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_{sr} & b_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_c & a_{sl}\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & b_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_c & \#\\\hline\end{array}$, $\begin{array}{|c|}\hline \#\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|}\hline b_i\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_i & b_i\\\hline\end{array}$,
$\begin{array}{|c|}\hline b_i\\\hline a_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_i & a_c\\\hline\end{array}$, $\begin{array}{|c|}\hline a_c\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_c & b_i\\\hline\end{array}$, $\begin{array}{|c|}\hline b_i\\\hline a_l\\\hline\end{array}$, $\begin{array}{|c|}\hline a_l\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_l & b_i\\\hline\end{array}$, $\begin{array}{|c|}\hline b_i\\\hline a_r\\\hline\end{array}$, $\begin{array}{|c|}\hline a_r\\\hline b_i\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_i & a_r\\\hline\end{array}$, $\begin{array}{|c|}\hline \#\\\hline a_i\\\hline\end{array}$, $\begin{array}{|c|}\hline a_i\\\hline a_i\\\hline\end{array}$,
$\begin{array}{|c|}\hline a_c\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|}\hline a_r\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & a_l\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_l & a_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_c & a_c\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_c & a_r\\\hline\end{array}$, $\begin{array}{|c|c|}\hline a_r & \#\\\hline\end{array}$, $\begin{array}{|c|}\hline a_l\\\hline a_{sr}\\\hline\end{array}$, $\begin{array}{|c|}\hline a_c\\\hline a_{sr}\\\hline\end{array}$, $\begin{array}{|c|}\hline a_r\\\hline a_{sl}\\\hline\end{array}$, $\begin{array}{|c|}\hline a_c\\\hline a_{sl}\\\hline\end{array}$,
$\begin{array}{|c|}\hline b_r\\\hline \#\\\hline\end{array}$, $\begin{array}{|c|c|}\hline \# & b_l\\\hline\end{array}$, $\begin{array}{|c|c|}\hline b_r & \#\\\hline\end{array}$, $\begin{array}{|c|}\hline b_l\\\hline a_{sr}\\\hline\end{array}$, $\begin{array}{|c|}\hline b_r\\\hline a_{sl}\\\hline\end{array}$ $|i \in \{l, c, r\}\}$.

The deterministic process starts in the lower left corner. If there is an a then a_{sr} is the only possible choice here since a_l can not occur over $\#$ and a_c and a_r can not occur right of $\#$. Then the right neighbor can only be a_{sr} since no other a_i can be right of an a_{sr}. This continues along the bottom line. Then the process proceeds on the right lower corner. If there is an a then a_{sl} is the only possible choice here since a_r can not occur over a_{sr} and a_c and a_l can not occur left of $\#$. Analogously the second line becomes a_{sl}. This continues in snakelike manner producing a_{sr} on the way right and a_{sl} on the way left until the first b is found, which is then forced to become a b_c since neither b_l nor b_r can be left of a_{sl} or right of a_{sr}. Then all connected b's must become b_c and all remaining a's become a_l, a_r or a_c depending on their position.

#	#	#	#	#	#	#	#	#
#	a_l	a_c	a_c	b_c	b_c	b_c	a_r	#
#	b_c	b_c	b_c	b_c	a_c	b_c	a_r	#
#	a_l	b_c	a_c	b_c	b_c	b_c	a_r	#
#	a_l	a_c	a_c	b_c	a_c	a_c	a_r	#
#	a_{sr}	a_{sr}	b_c	b_c	a_c	a_c	a_r	#
#	a_{sl}	a_{sl}	a_{sl}	a_{sl}	a_{sl}	a_{sl}	a_{sl}	#
#	a_{sr}	a_{sr}	a_{sr}	a_{sr}	a_{sr}	a_{sr}	a_{sr}	#
#	#	#	#	#	#	#	#	#

#	#	#	#	#	#	#	#	#
#	a_l	a_c	a_c	b_c	a_c	b	a_r	#
#	b	a_c	b_c	b_c	a_c	b	a_r	#
#	a_l	b_c	a_c	b_c	a_c	b	a_r	#
#	a_l	a_c	a_c	b_c	a_c	a_c	a_r	#
#	a_{sr}	a_{sr}	b_c	b_c	a_c	a_c	a_r	#
#	a_{sl}	a_{sl}	a_{sl}	a_{sl}	a_{sl}	a_{sl}	a_{sl}	#
#	a_{sr}	a_{sr}	a_{sr}	a_{sr}	a_{sr}	a_{sr}	a_{sr}	#
#	#	#	#	#	#	#	#	#

But if the b's are not connected, then some of them can not be determined to b_c, b_l or b_r and the process stops, as shown in the right picture. Note that the tiling is not monocausal since an a left of a b_c can only become a a_c if the a under that a became an a_c or a_{sr} (and not an a_{sl}); but it could be made monocausal by introducing two more b_i-symbols for the first b.

The fact that $\mathcal{L}_{md}(\Delta, \pi)$ and $\mathcal{L}(\Delta, \pi)$ might be different makes the following non trivial:

Theorem 3. *Every monocausal deterministically recognizable language is recognizable.*

Proof. (Sketch) The idea is a generalization of the tentacle method in the proof of Theorem 1. The tree which was used there corresponds to the order of the replacements in Theorem 2: A b was replaced by b_c if the 'parent' b had been replaced by b_c before. Every cell contains encoded *tentacles* like in Theorem 2 and additional the images of the 4 neighbors (which is checked by the tiling) and one third pointer. These third pointers use the same ways (but not the same direction) as the causal pointers and connect all cells to a forest rooted to #'s, which together with the tentacle forest guarantees the cycle freeness. The tilings simulate the monocausal replacement.

Conjecture Every deterministically recognizable language is recognizable.

What changes in the general case is that several (up to 4) neighbors together can force one cell to be replaced, which means instead of a tree we need a planar directed acyclic graph to simulate the order of replacements in the deterministic process.

Open problem Are the deterministically recognizable languages closed under complement?

4 Nonuniform Counting

Nonuniformity is a widely used principle in theoretical computer science. It says that we do not need one algorithm, Turing machine, grammar or whatever to recognize a language but we may use a hole family of them, where each is used only for words of one special size. Connections of nonuniformity and counting can for example be found in [RA97] and [AR98]. A common characterization of nonuniformity is by advice strings. One major observation is that most lower bounds of problems or statements saying that a problem does not belong to a certain class also hold for the nonuniform version of the measure or class. This also holds for Lemma 1. It is easy to see that, as long as we keep the size of the alphabet constant, the lemma does not make use of uniformity. It is an open problem, whether the language of pictures over $\{a, b\}$, where the number of a's is equal to the number of b's and having a size (n, m) with $\log n \leq m \leq 2^n$ is recognizable. The following result shows that it is not possible showing its non recognizability using Lemma 1.

Definition 3. *For 2 pictures $p \in \Sigma^{*,*}$ and $q \in \Gamma^{*,*}$ of size (m, n) the product $p \times q \in (\Sigma \times \Gamma)^{*,*}$ is defined by $(p \times q)(i, j) = (p(i, j), q(i, j))$.*

A picture language $L \subseteq \Sigma^{*,*}$ is called nonuniformly local *if there is an infinite 2-dimensional array of advice-pictures ($a_{m,n} \in \Gamma^{*,*}$) and a local picture language $L' \subseteq (\Sigma \times \Gamma)^{*,*}$ with*

$$p \in L \Leftrightarrow p \times a_{m,n} \in L'$$

for every picture p of size (m, n).

Theorem 4. *The language of pictures over $\{a, b\}$, where the number of a's is equal to the number of b's and having a size (n, m) with $\log n \leq m \leq 2^n$ is nonuniformly recognizable.*

For the proof we need the following closure property:

Definition 4. *A horizontal (or vertical)* folding *is a function $f : \Sigma^{*,*} \mapsto (\Sigma^2)^{*,*}$, which maps a picture $p \in \Sigma^{*,*}$ of size $(m, 2n)$ (or $(2m, n)$) to $f(p)$ of size (m, n) with $(f(p))(i, j) = (p(i, j), p(i, 2n - 1 - j))$ (or $(f(p))(i, j) = (p(i, j), p(2m - 1 - i, j))$). For a picture language L we define the folding $f(L) = \{f(p) | p \in L\}$.*

Lemma 3. *The (nonuniformly) recognizable languages are closed under folding.*

Proof. (Sketch of Theorem 4) Because of the last lemma it suffices to restrict to those cases, where we only have to count the difference of a's and b's in the upper left quadrant (we may for example assume the rest to be filled with alternating stripes of a's and b's); by a finite number of foldings and projecting to the upper left quadrant we get the original language. Furthermore w.l.o.g. we assume the width to be greater than the height.

The essential idea of the proof is that the counter is constructed from small constant size counters which have different order. The orders are powers of 2. The number of occurrences of a counter with order 2^{2i} is exponentially decreasing with i similar to the counter used in [Für82]. Since the order can not be known on the local level (the alphabet is finite but not the order), the advice is needed to tell, when counters can be combined.

The constant size counters of a column represent a counter state, which holds the difference of a's and b's left of this column. We simulate a process moving the counter from left to right performing an increment for each a and a decrement for each b. Let every cell c have 3 counters c_1, c_2, c_3, with $-2 \leq c_i \leq 2$. The order of the first counter is always 1, the order of the second counter depends only on the row:

- In the upper half the order is 2^{2i} in all rows $2^{i-1} + j2^i$ for every i, j.
- In the lower half the order is 2^{2i} in the row i for every i.

The order of the third counter is 2^{2i-1} if column + n - row= $2^{i-1} + j2^i$, where n is the height of the picture. A cell c has an advice $c_a \in \{0, s, g, 2, 4\}$ and an effect $c_e \in \{-1, +1\}$ which is $c_e = 1$ if $\pi(c) = a$ and $c_e = -1$ if $\pi(c) = b$, this means an a increments the counter and a b decrements the counter. An example for the order of the counters is the following:

1	4	2	1	4	512	1	4	2
1	16	8	1	16	2	1	16	512
1	4	2	1	4	8	1	4	2
1	64	32	1	64	2	1	64	8
1	4	2	1	4	32	1	4	2
1	16	8	1	16	2	1	16	32
1	4	2	1	4	8	1	4	2
1	256	128	1	256	2	1	256	8
1	4	2	1	4	128	1	4	2
1	16	8	1	16	2	1	16	128
1	64	2	1	64	8	1	64	2
1	256	32	1	256	2	1	256	8
1	1024	2	1	1024	32	1	1024	2
1	4096	8	1	4096	2	1	4096	32
1	16384	2	1	16384	8	1	16384	2
1	65536	512	1	65536	2	1	65536	8

with the advice

4	0	4
g	2	0
4	s	4
g	2	0
4	0	4
g	2	s
4	s	4
g	2	0
4	0	4
g	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

We allow tiles of the form $\begin{array}{|c|c|}\hline \mathrm{c} & \mathrm{d} \\ \hline \mathrm{e} & \mathrm{f} \\ \hline \end{array}$ in the following 5 cases:

- $f_a = 0$, $e_1 + f_e = f_1$, $e_2 = f_2$, $c_3 = f_3$, which means the first counter has to take the effect, the second counter is just moved to the next column and the third counter is moved to the next column simultaneously changing the row
- $f_a = s$, $e_1 + f_e = f_1$, $e_2 + 2c_3 = f_2 + 2f_3$, which means additionally the second counter has the double order of the third counter, which allows transfer between them
- $f_a = g$, $e_1 + f_e = f_1$, $2e_2 + c_3 = 2f_2 + f_3$, which means the second counter has the half order of the third counter
- $f_a = 2$, $e_1 + f_e + 2c_3 = f_1 + 2f_3$, $e_2 = f_2$, which means the first counter has the half order of the third counter

– $f_a = 4$, $e_1 + f_e + 2c_3 + 4e_2 = f_1 + 2f_3 + 4f_2$, which means the first counter has the half order of the third counter and the forth order of the second counter.

A counter of a certain order will meet two times a counter of half order and take their load until it meets a counter of double order, where it can get rid of its load. If it has for example a 1, then it can nondeterministically decide to keep it or to get -1 and increment the counter with double order by 1. Third counters are only allowed leaving the picture at the bottom if they are zero. A picture is in the language iff the tiling system can simulate a process of a counter starting at the leftmost column with zero and ending at the rightmost column with zero.

Acknowledgment: We thank V. Diekert, H. Fernau, K.-J.Lange, P. McKencie, O. Matz, W. Thomas and T. Wilke for helpful discussions.

References

[AR98] E. Allender and K. Reinhardt. Isolation matching and counting. to appear in Proc. of 13th Computational Complexity, 1998.

[FSV95] Ronald Fagin, Larry J. Stockmeyer, and Moshe Y. Vardi. On monadic NP vs. monadic co-NP. *Information and Computation*, 120(1):78–92, July 1995.

[Für82] Martin Fürer. The tight deterministic time hierarchy. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 8–16, San Francisco, California, 5–7 May 1982.

[GR96] D. Giammarresi and A. Restivo. Two-dimensional languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III. Springer-Verlag, New York, 1996.

[GRST94] Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over pictures and recognizability by tiling systems. In P. Enjalbert, E.W. Mayr, and K.W. Wagner, editors, *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science, STACS 94 (Caen, France, February 1994)*, LNCS 775, pages 365–375, Berlin-Heidelberg-New York-London-Paris-Tokyo-Hong Kong-Barcelona-Budapest, 1994. Springer-Verlag.

[IN77] K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*, 13:95–121, 1977.

[Mat97] Oliver Matz. Regular expressions and context-free grammars for picture languages. In *14th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1200 of *lncs*, pages 283–294, Lübeck, Germany, 27 February–March 1 1997. Springer.

[Mat98] Oliver Matz. On piecewise testable, starfree, and recognizable picture languages. In Maurice Nivat, editor, *Foundations of Software Science and Computation Structures*, volume 1378 of *Lecture Notes in Computer Science*, pages 203–210. Springer, 1998.

[RA97] K. Reinhardt and E. Allender. Making nondeterminism unambiguous. In *38 th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 244–253, 1997.

[Wil97] Thomas Wilke. Star-free picture expressions are strictly weaker than first-order logic. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming*, volume 1256 of *Lect. Notes Comput. Sci.*, pages 347–357, Bologna, Italy, 1997. Springer.