Ville Salo

Classes of Picture Languages Defined by Tiling Systems, Automata and Closure Properties

Turku Centre *for* Computer Science

TUCS Technical Report
No 1007, April 2011

CLASSES OF PICTURE LANGUAGES DEFINED BY TILING SYSTEMS, AUTOMATA AND CLOSURE PROPERTIES

Ville Salo

Master's Thesis
March 2011

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF TURKU
FIN-20014 TURKU
FINLAND

UNIVERSITY OF TURKU
Department of Mathematics

SALO, VILLE: Classes of picture languages defined by tiling systems, automata and closure properties
Master's thesis, 91 pages, 2 appendices
Mathematics
March 2011

This thesis is about classes of picture languages, relations between them and languages they do and do not contain. The theory of picture languages is a lesser-known branch of formal language theory that studies languages of finite two-dimensional matrices called pictures. Many models have been introduced for defining picture languages, and in this thesis we investigate, in depth, classes of picture languages defined by tiling systems, automata and closure properties. In addition to results about the kinds of languages contained in each class, we have aimed for a relatively complete presentation of the interesting relations between the classes. In particular, we have tried to include the most interesting proof techniques used in the field. We give detailed and complete proofs for most of the theorems we state, usually with an original proof.

The thesis contains many new results, and many answers to questions asked in the literature. The most interesting of these is the proof that (picture-walking) nondeterministic finite state automata run on pictures give the same class of languages whether or not they are allowed to exist the domain of the picture. A similar question, also asked in the literature, of whether the same is true for alternating finite state automata, is answered in the other direction. We also prove that the class defined by so-called right linear grammars is a subclass of the class of picture languages defined by north-west deterministic tiling systems, which was asked in the Handbook of Formal Languages. We also prove many other natural theorems not previously known, for instance the inclusion of the class of locally threshold testable languages to the class defined by north-west deterministic tiling systems.

Keywords: picture languages, tiling systems, automata, closure properties.

Contents

1 Introduction

1.1 General approach

In this thesis, the theory of picture languages, a lesser-known branch of formal language theory, is built with special focus on definitional issues such as the role of the border of a picture, closure properties of picture classes and relations between classes. We give solutions to several problems asked in the literature and give novel proofs for many results already known.

As there already exist a huge amount of models for defining classes of picture languages, while in many cases a basic theory seems to be lacking, exploring new models of computation seems rather pointless. Instead, new variations of existing models are explored in an effort to fill some of the gaps in the existing theory.

1.2 Background

Informally, a picture is a matrix over a finite alphabet, and a picture language is a set of matrices over the same alphabet. The theory of picture languages is a branch of formal language theory, and as such doesn't concentrate on combinatorial aspects of pictures, but instead on the structure of, and relations between, *classes of languages*, sets of picture languages usually defined in some natural way.

The idea of extending formal language theory to two dimensions is an old one, picture languages appeared already by the end of 1960's, motivated by problems in pattern recognition and image processing [1]. In the literature, countless formalisms have been introduced for defining classes of picture languages. The three most important families of formal systems considered in this thesis are those defined by

- assigning states to all cells of a picture, and checking that a local rule holds everywhere in the configuration obtained.
- an automaton moving on the positions of the picture, one cell at a time.
- closure properties and generating devices.

The idea of assigning states to all cells, and checking a local rule, was first introduced in [3], where the class of picture languages accepted by so-called 'online tessellation automata' was defined. The same class of languages was later given a much nicer definition in [2], presumably inspired by the theory of sofic shifts. The second idea of 'picture-walking' automata, finite state automata moving in all four directions on the cells of the picture, was introduced already in 1967 in [4]. Closure properties have been studied at least in the context of regular expressions, for instance in [5].

Not surprisingly, many problems in the theory of picture languages originate from the theory of one-dimensional languages. For instance, the logical characterization of [6] for the recognizable languages is similar to the classical Büchi's theorem [21]. However, while giving surprising new characterizations of the regular languages is a recurring theme in one-dimensional formal language theory, the two-dimensional case turns out to behave less nicely, and incomparability results seem to be more frequent than nontrivial inclusion results.

1.3 Structure of the thesis

Chapter 2 defines one of the most important classes of picture languages, the class of recognizable picture languages. We give simple examples of languages in this class, and some of its well-known closure and nonclosure properties. Recognizable relations are introduced as a new technique for proving a language is in REC. As an application, we show REC is closed under the tiling operation that composes pictures from smaller pictures. Matz' lemma for showing a language is not recognizable is proven for general graphs, and we directly obtain two ways to use it for pictures. We give a language theoretic proof for the results LTT = FO and REC = EMSO, found in [6]. In the last section, we build a (presumably) well-known inclusion lattice out of LOC, LTT, DREC and REC, and compare DREC with its closure under rotations.

In Chapter 3, we consider picture-walking automata, finite state automata that move on the positions of the picture according to a transition rule, accepting the picture if they reach a final state. Basic closure properties and nonclosure properties are proven. We solve two problems asked in [7], the relationship between the classes AFA and FAFA, and the relationship between classes NFA and FNFA. In order to prove the proper inclusion of AFA in FAFA, we need the result that alternating one-dimensional two-way finite state automata accept exactly the regular languages. We give a straightforward proof for this well-known theorem, since there do not seem to be many proofs for it in the literature. Also, the arguments of [7] proving certain properties of the classes DFA, NFA and AFA are extended in a straightforward way for the class UFA allowing only universal quantification in states. We discuss the possible limits of UFA, and in particular conjecture that UFA lacks one of the most basic closure properties: being closed under union.

Chapter 4 is about picture language classes defined by closure properties, in particular certain types of regular expressions for defining picture languages are given. We solve the problem of whether there exists a relationship between DREC and RLG, which was asked in [1], and give a new proof for the incomparability of RE and REC, which was also asked in [1] but already has a published solution in [5] (which we were unaware of). RLG is also given a new characterization using certain types of restricted automata.

2 Recognizable picture languages

In the theory of one dimensional languages, the regular languages have a plethora of characterizations. In this chapter, we generalize the following characterization of the regular languages to two dimensional words (matrices over a finite alphabet which we call 'pictures').

Definition 2.1. A *local one-dimensional language* is a language defined by a set of uniform local constraints. More precisely, with each tuple (α, R, ω) where $\alpha, \omega \subset \Sigma$ and $R \subset \Sigma \times \Sigma$ for a finite set Σ, we associate the language $\{w \in \Sigma^* \mid w_1 \in \alpha, w_{|w|} \in \omega, \forall\ 1 \leq i \leq |w| - 1 : (w_i, w_{i+1}) \in R\}$. *Recognizable one-dimensional languages* are the languages obtained from local languages through letter-to-letter substitutions.

The class REC of recognizable *two dimensional* languages (picture languages) was defined in the article [6] by D. Giammaresi and A. Restivo, and defines the same languages as an earlier computation model called on-line tessellation automata, defined in the article [3]. We formulate some of the basic theory of recognizable picture languages already known, and make a few simple additions to this theory. In particular, we introduce a new 'programming technique' for defining recognizable picture languages. This technique formalizes an idea already extensively used in many constructions, but never explicitly stated, as far we know.

In this thesis, the class REC is considered the most natural 'finite state' class in the theory of picture languages, for reasons scattered all around the thesis. We will discuss the closure properties of this class in some detail, and later, after defining other classes, this class provides a natural point of comparison. Often, we will take a handful of language classes, compare them with respect to set inclusion and draw a Hasse diagram depicting the relations in compact form. In these diagrams, REC will always take part.

2.1 Definitions and basic results

In this section, basic definitions are given, and the picture classes LOC and REC are defined. We define the local languages LOC using arbitrarily large neighborhoods, and define REC as the symbol-to-symbol projections of languages in LOC. We then work our way to a characterization of REC as the projections of local languages with one of two fixed neighborhoods, the neighborhood of domino tiles or the square tile neighborhood.

Because we will often build classes of picture languages containing certain languages over Σ for arbitrary alphabet Σ, it will make our life easier if we assume all alphabet symbols are from an anonymous countable set that contains, among other things, most of the symbols, numbers and finite sets man will ever use in mind and on paper, and that is closed under all sensible set operations such as cartesian products and unions. Some exceptions are $\#$, $\#'$, and in general every symbol that we give a special meaning to. If no restriction for the alphabets were given, these 'classes' would have to be proper classes.

Also all kinds of colored, dotted and striped squares are perfectly valid alphabet letters (physical chess pieces might be pushing it though), and we will often just give drawings without assigning symbols to the colors. Sometimes,

though, we will use colors in pictures, and symbols in the text, to represent the same symbol. Things like this will always be explained in the text.

Alphabets themselves need not be just sets, but may be the finite 'set part' of a more complicated object such as a group. For instance, we could talk about pictures over a group or over an ordered set.

Without further ado, we now start with basic definitions, first defining pictures and their languages.

Definition 2.2. Let Σ be a finite set called the *alphabet* and let $\#$ be a special symbol not in any alphabet considered. A *picture* p over Σ is a pair (Σ, c) where $c \in (\Sigma \cup \{\#\})^{\mathbb{Z}^2}$ has the property that the set of positions of c containing an element of Σ is a finite rectangle, which we refer to as the content of the picture. Pictures over Σ can be thought of as matrices over Σ with a special symbol $\#$ everywhere outside the matrix.

Note that there is only one empty picture, while in theory one might define empty pictures of all sizes $0 \times n$, $n \times 0$. This detail will be significant when defining regular expressions later.

Definition 2.3. The set of all pictures over Σ is denoted by Σ^{**}. A *picture language* L over Σ is a subset of Σ^{**} closed under translations of pictures. That is, the if p is in L, then also every other picture containing the same rectangle over Σ surrounded by $\#$ is in L. We denote by $\mathrm{alph}(L)$ the set of symbols (not including $\#$) actually used in the pictures of L.

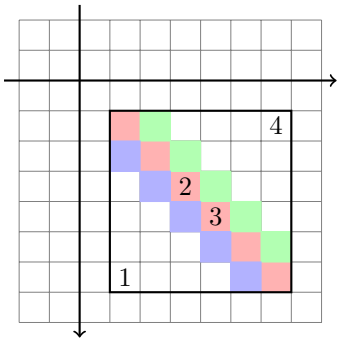
Definition 2.4. A *picture class* CLS is a set of picture languages closed under bijective alphabet substitutions. That is, it is a set of picture languages with the additional property that if $L \in \mathrm{CLS}$, where $L \in \Sigma^{**}$, then if $f : \Sigma \to \Sigma'$ is a bijection, then necessarily the language $\{f(p) \mid p \in L\}$ obtained by substituting letters of Σ' for letters of Σ is also in CLS.

Note that a picture class will always contain picture languages with all possible symbols (unless it is empty), but a picture language will always be over exactly one finite alphabet.

We could also have defined languages as tuples (Σ, X) where Σ gives the alphabet used, and X is a subset of Σ^{**}. However, defining the alphabet of a language 'intrinsically', by just looking at the actual pictures, is nice, because it means a language doesn't have to carry information about its alphabet, and two languages are the same if and only if they have the same pictures. However, we will occasionally think of the alphabet of a language as containing symbols not actually used in the pictures of the language. That is, *the alphabet* Σ *of* L, as given by $\mathrm{alph}(L)$, is always the unique alphabet whose symbols L uses, but it is also true that L *is a language over* Σ' for finite superalphabets Σ' of Σ. This detail is sometimes important, since certain language operations constrain the cardinality of the language.

The property of being closed under substitutions forbids classes from having any kind of structure in their alphabets. It has very important consequences when defining picture classes by their closure properties, and we will elaborate on this in Section 2.2. When defining classes in other ways, it will usually be obvious that they are closed under substitutions, and we will not explicitly check this. Another thing implied by the property is that we can think of two languages obtained from each other using a symbol-to-symbol bijection as being

Figure 1: Illustration of a typical picture at a typical position on the plane.



'essentially the same'. (In fact, just like the orbit of a picture moves the picture everywhere on the plane, giving us infinitely many views of the same picture, we can think of languages as having orbits, with symbol-to-symbol bijections giving the 'dynamics'.)

We define pictures as configurations of the whole plane instead of directly defining them as matrices for two reasons. First of all, this makes 'going outside the picture', which we will often need to do, more natural, and gives a direct connection between picture languages and the theory of two-dimensional shift spaces, although we will only talk about this topic briefly, in Section 2.3. The second reason is more technical, and less convincing: letting us move pictures around makes things like juxtapositioning slightly easier and more natural to define, since we have one thing less to worry about when even non-rectangular gluings of pictures give well-defined mathematical objects (although not necessarily pictures).

Because pictures are essentially matrices, *we will always think of the first axis of* $\mathbb{Z}^2$ *as being vertical and of the second axis as being horizontal* in this thesis. Also, *the positive direction of the first axis of* $\mathbb{Z}^2$ *extends downwards from the origin*, and the positive direction of the second axis extends to the right. Note that the plane is usually not drawn this way.

When we say 'given a picture' in a proof, we will usually assume said picture has its content in a rectangle of the form $[1, ..., m] \times [1, ...n]$, and when defining languages, we will not explicitly mention taking the closure under translations. Our pictures, when in this 'usual' position, do not touch the origin, but leave a small gap around the axes (see Figure 1). A rationalization for this might be that the #-cells immediately next to the content of the picture form a kind of border for the picture, and this bordered version of the picture forms an initial lower right segment of the plane starting from the origin.

Figure 1 is an illustration of a typical picture at a typical position on the plane. We draw white cells instead of # outside the picture, to avoid the needless distraction. We also draw a rectangular border around the picture, even though the definition of picture doesn't talk about a border. The interpretation is that white cells outside the bordered area denote #, while white cells within it are simply empty white cells, a valid alphabet symbol.

We now give some technical definitions to make precise definitions and formal

proofs more concise.

Definition 2.5. Let $\mathcal{B} = \{b \subset \mathbb{Z}^2 \mid |b| < \infty\}$ be the set of *blocks*. When $b_1, b_2 \in \mathcal{B}$, we write $b_1 \sim b_2$ if $\exists x : b_1 + x = b_2$. When $b \in \mathcal{B}$, an element of Σ^b is called a *tile* over Σ of shape b. We denote the set of all such tiles by $\mathcal{T}_\Sigma$. When $t_1 \in \Sigma^{b_1}$ and $t_2 \in \Sigma^{b_2}$, we write $t_1 \sim t_2$ if $\exists x : b_1 + x = b_2$ and $\forall y : t_1(y) = t_2(y + x)$. The equivalence relation $\sim$ is called *similarity*.

In $\mathcal{T}_\Sigma$, the subscript Σ is often omitted when it is clear from the context. Usually, it is then assumed to be Σ. Tiles are also called patterns, and the two terms are used interchangeably.

In the following definition, we give some ways to extract information about the size of a picture, and its location on the plane.

Definition 2.6. If $p \in \Sigma_*^*$ and p is not the empty picture, we define the top, bottom, left and right edge of picture p respectively as

$$\operatorname{top}(p) = \min_i \exists j : p[i, j] \in \Sigma,$$

$$\operatorname{bottom}(p) = \max_i \exists j : p[i, j] \in \Sigma,$$

$$\operatorname{left}(p) = \min_j \exists i : p[i, j] \in \Sigma,$$

and

$$\operatorname{right}(p) = \max_j \exists i : p[i, j] \in \Sigma,$$

respectively. Note that these are all well-defined if p is nonempty. If p is the empty picture, $\operatorname{top}(p) = \operatorname{left}(p) = 1, \operatorname{bottom}(p) = \operatorname{right}(p) = 0$. We define the offset of p as $\operatorname{offset}(p) = (\operatorname{top}(p) - 1, \operatorname{left}(p) - 1)$, the width of p as $|p| = \operatorname{right}(p) - \operatorname{left}(p) + 1$ and the height of p as $\underline{\overline{p}} = \operatorname{bottom}(p) - \operatorname{top}(p) + 1$. We define the domain of p as

$$\operatorname{dom}(p) = \{(i, j) \mid \operatorname{top}(p) \leq i \leq \operatorname{bottom}(p), \operatorname{left}(p) \leq j \leq \operatorname{right}(p)\},$$

its extended domain as

$$\operatorname{edom}(p) = \{(i, j) \mid \operatorname{top}(p) - 1 \leq i \leq \operatorname{bottom}(p) + 1, \operatorname{left}(p) - 1 \leq j \leq \operatorname{right}(p) + 1\},$$

and the corners of p as

$$\begin{aligned}\operatorname{corners}(p) = \{&(\operatorname{top}(p), \operatorname{left}(p)), (\operatorname{top}(p), \operatorname{right}(p)),\\ &(\operatorname{bottom}(p), \operatorname{left}(p)), (\operatorname{bottom}(p), \operatorname{right}(p))\}.\end{aligned}$$

The shape of picture p is the vector

$$\operatorname{shape}(p) = (\underline{\overline{p}}, |p|).$$

The shape $\operatorname{shape}(p)$ of picture p is especially important in the case of a unary language L, since in this case the set $\operatorname{shape}(L)$, that is, the shapes of the pictures in L, completely determines L.

We already briefly discussed the fact that since all translates of a picture always occur together in languages, we can usually just take a single representative of each picture. The following gives a name to this representative.

Definition 2.7. A picture p is said to be in *normal form* if $\text{offset}(p) = (0,0)$. We identify pictures in normal form over Σ with matrices over Σ.

The reason we subtract 1 on both axes in the definition of offset is that we're interested in the offset from the left upper corner of pictures in normal form, which is $(1,1)$.

Let us now give a set of indexing functions, that is, ways to extract parts of the content the picture.

Definition 2.8. We write $p[i,j]$ for the (i,j)th cell of p, $p[i,*]$ for the ith row of p, that is, the word of length $|p|$ defined by $p[i,*]_j = p[i,j]$. Similarly, $p[*,j]$ denotes the jth column of p, read top down. If $b \in \mathcal{B}$, we let $p[b] = t$ such that $t \in \Sigma^b$, and $\forall x : t(x) = p[x]$. The content of p is the matrix $\text{cont}(p) = p[\text{dom}(p)]$. We also define the occurrence functions

$$o(p,b) = \{t \in (\Sigma \cup \{\#\})^b \mid \exists x : p[b+x] \sim t\},$$

$$o(p) = \bigcup_{b \in \mathcal{B}} o(p,b),$$

and the count function

$$c(p,t) = |\{x \in \mathbb{Z}^2 \mid p[x + \text{dom}(t)] \sim t\}|,$$

where $|X| = \infty$ if X is infinite.

Note that while pictures in normal form can be identified with matrices, we have a natural one-to-one correspondence between matrix-offset pairs and pictures in general.

The following definition explains how a symbol-to-symbol function is lifted into a function from pictures to pictures, and from languages to languages. This also gives meaning to expressions such as $\text{shape}(L)$.

Definition 2.9. Let $f : A \to B$ and let X be a subset of A. Then we define $f(X) = \{f(x) \mid x \in X\} \subset B$. If $f : \Sigma \to \Delta$ and $p \in \Sigma_*^*$, then $f(p)$ is defined by $\forall i,j : f(p)[i,j] = f(p[i,j])$, and it is a picture over Δ.

We now get to the point, and define our first, and most important picture classes. We define local languages with purely local rules, using sets of forbidden blocks. We then define the recognizable languages by erasing information from the pictures of local languages. We may think of the local language as describing computations, and the recognizable language as describing the results of those computations.

Definition 2.10. A *local grammar* G is given by a pair (Σ, T) where Σ is a finite alphabet and $T \subset \mathcal{T}_{\Sigma \cup \{\#\}}, |T| < \infty$. The *local language* $\mathcal{L}(G)$ associated with G is defined as $\{p \in \Sigma_*^* \mid \nexists\, b \in \mathcal{B}, t \in T : t \in o(p,b)\}$. The set of all local languages is denoted by LOC.

It should be pointed out that with our definition, every nonempty local language contains the empty picture.

Definition 2.11. The class REC of *recognizable languages* is defined as the set $\{f(L) \mid L \in \text{LOC}, \exists \text{ alphabet } \Sigma : (f : \text{alph}(L) \to \Sigma)\}$.

REC is also clearly the smallest class of picture languages that contains the local languages and is closed under symbol-to-symbol projections, since such a class will necessarily contain REC, and symbol-to-symbol projections are closed under composition.

We could (but won't!) also define recognizable picture languages by a grammar with the signature $G = (\Sigma, \Delta, f^R, T)$ where $f : \Delta \to \Sigma$, and have the recognizable language $\mathcal{L}(G)$ associated with G be $f(\mathcal{L}((\Delta, T)))$ for the local grammar (Δ, T). This is what one would directly obtain by, for each REC language L, making a tuple out of all the information needed for producing L: the local language and the projection used.

Instead, we take a slightly different approach to REC grammars, generalizing the idea of the previous paragraph. We can, by definition, think of the language associated with an REC grammar G as being generated by taking locally consistent pictures over the intermediate alphabet, and taking their symbol-to-symbol images using some function f. However, another useful way is to think of an REC grammar as a description of a process recognizing a picture language. The process is given a picture over Σ, and it tries to guess a symbol of Δ called a *state* at each position on the other layer of the cartesian product, respecting the local constraints. That is, it guesses a preimage for the given picture. Of course, the term 'recognizable picture language' makes more sense using this interpretation.

There is, however, a slight inconsistency with this interpretation and the definition of REC: since we are guessing a preimage through a function f, the state of a cell must uniquely determine the symbol of the image in that cell. Sometimes, it would be more natural to let the same state be assigned on top of multiple symbols. To this end, we give the following definition.

Definition 2.12. A *recognizable grammar* is a tuple $G = (\Sigma, \Delta, R, T)$, where $R \subset \Sigma \times \Delta$ and T is a set of tiles over Δ. The language $\mathcal{L}(G)$ is defined as

$$\{p \in \Sigma_*^* \mid \exists p' \in \Delta_*^* : \mathrm{dom}(p) = \mathrm{dom}(p'), \forall (i,j) : p[(i,j)]\, R\, p'[(i,j)] \wedge p' \in \mathcal{L}((\Delta, T))\},$$

where (Δ, T) is a local grammar.

In the proofs, we will refer to R as the *state assignment rule* and to T as the *local rule*. The state assignment rule gives the set of Δ symbols allowed on top of each symbol of Σ, and the local rule describes the allowed and forbidden local patterns over Δ.

It should be clear that recognizable grammars have their language in REC, since we can always find an equivalent grammar where R is a function by adding symbol information to states.

Definition 2.13. Let $B \subset \mathcal{B}$ be finite. The local grammar (Σ, T) is of the *type* B if $\{\mathrm{dom}(t) \mid t \in T\} \subset B$ modulo $\sim$-equivalence. The recognizable grammar (Σ, Δ, R, T) is of type B if the local grammar (Δ, T) is. Note that a grammar can be of multiple types. We say a local or recognizable language is of type T if there is a grammar of type T for it.

Definition 2.14. We say b_1 is a *subblock* of b_2 is $\exists x : b_1 + x \subset b_2$. We define superblocks, subtiles and supertiles similarly, in the obvious way.

The main result Theorem 2.19 of this chapter is about finding a type B such that all languages in REC are of type B. Let us first show that a similar

result can not be proven for LOC. The following characterizes when one local grammar type can simulate all local grammars of another type.

Theorem 2.15. *Let B_1 and B_2 be finite subsets of $\mathcal{B}$. Then the following are equivalent:*

(a) For every local grammar G_1 of type B_1 there is a local grammar G_2 of type B_2 such that $\mathcal{L}(G_1) = \mathcal{L}(G_2)$.

(b) $\forall b \in B_1 : \exists c \in B_2 : b$ is a subblock of c.

Proof. Assume (b). Consider an arbitrary grammar G_1 of type B_1 with forbidden tiles T. For each $t \in T$ we can find a block $b \in B_2$ such that $\mathrm{dom}(t)$ can be extended to a block c similar to b. By filling the new positions of c into a supertile of t in all possible ways we obtain a grammar of type B_2 with the language $\mathcal{L}(G_1)$. This proves (b) $\Rightarrow$ (a).

Assume (a) is true, but (b) is false. Then there is a block $b \in B_1$ that doesn't occur as a subblock of any block of B_2. Define the grammar $G_1 = (\{0,1\}, \{1\}^b)$, and let $L = \mathcal{L}(G_1)$. Let G_2 be a grammar of type B_2 with forbidden tiles T_2 such that $\mathcal{L}(G_2) = L$. Let p be a picture of all zeroes except for a subblock c of ones such that $c \sim b$. Now $p \notin L \Rightarrow p \notin \mathcal{L}(G_2)$, so there must exist a block d and a tile $t \in T_2$ such that $p[d] \sim t$. But there must be a position x such that $p[x] = 1 \wedge x \notin d$. Define a new picture q by

$$q[y] = \begin{cases} p[y] & \text{if } y \neq x \\ 0 & \text{if } y = x \end{cases}$$

Then $q[d] = p[d]$, so $q \notin \mathcal{L}(G_2)$. But clearly $q \in L$, a contradiction. This proves (a) $\Rightarrow$ (b). □

In the computation analog, the previous theorem tells us that local rules with bigger neighborhoods can compute 'faster', or at least send signals further away. It turns out, however, that under relatively modest assumptions on the shape of the neighborhood, 'universal computation', in terms of the result of the computation, is achieved. Similarly, the computations of Turing machines whose heads can jump 5 cells at a time look different from those of normal Turing machines, but when the information about the computation steps taken to produce the results are removed, we get the same classes of languages. In particular, in Theorem 2.19 we show that if the natural basis of $\mathbb{Z}^2$ is contained in a set of blocks, we can use this set of blocks as the local language of any recognizable language.

Definition 2.16. Let $B \subset \mathcal{B}$. The *signal set* S of B is $\{x - y \mid b \in B, \{x, y\} \subset b\}$.

The following lemma will be useful in the proof of Theorem 2.19.

Lemma 2.17. *Let $B \subset \mathcal{B}$ and G a recognizable grammar of type B. Then there is a recognizable grammar of type $\{\{x, y\} \mid b \in B, \{x, y\} \subset b\}$ with the same language. We say such a type is the* thinning *of type B. Conversely, for every language L implementable with the thin type, L can also be implemented with type B.*

Proof. It is enough to show the first claim for local grammars, since REC is closed under projections, without changing the type.

Let S be the signal set of B. We take Σ^S as the set of states. At each position, the interpretation of the state s at position x is that

$$\forall v \in S : s(v) = a \iff p[x + v] = a,$$

where q is some configuration of states proving $p \in L$.

The state assignment rule is that state s of a position containing symbol a must have $s((0,0)) = a$, and the local rules is that for each $v \in S$, tile t of shape $\{0, v\}$ is forbidden if

$$t(0)(v) \neq t(v)((0,0)),$$

with the interpretation that $s((0,0)) = \#$ for the state s of the special symbol $\#$.

Then, each state will know the symbols in the neighbors a signal can reach, so all we need to do is forbid states containing a forbidden tile: Then, for a picture $p \in L$, we find a consistent set of states for the positions by taking the allowed pattern around the position. Conversely, if a picture contains a forbidden pattern t, then at least one position $x \in \mathrm{dom}(p)$ is in this pattern, and its local state would have to contain t as well. Therefore the language of the new recognizable grammar is exactly L.

The latter claim is a direct corollary of Theorem 2.15, since we may implement the local language of a grammar over the thinning using the original type. □

Let us formally define universal local grammar types, capable of expressing any recognizable language.

Definition 2.18. We say a local grammar type B is *universal* if for all $L \in \mathrm{REC}$, there is a recognizable grammar G' of type B with $\mathcal{L}(G') = L$.

Theorem 2.19. *A local grammar type B is universal if and only if for each vector v in the natural basis $\{(0,1),(1,0)\}$ of $\mathbb{Z}^2$, there is a superblock $b \in B$ of the block $\{(0,0), v\}$.*

Proof. Again, to show such a local grammar type B of this form is universal, it is enough to show that every local language can be implemented as a recognizable language of this type. So let L be a local language with grammar G, and let B' be a grammar type such that each vector of the natural basis is a subblock of some block in B'. We need to prove there exists a recognizable grammar of type B' with language L. By Lemma 2.17, it enough to show there exists a grammar G' for L with tiles of shapes B'' for the thinning B'' of B'. We note that the blocks $N' = \{(0,0)\} \times \{(0,1),(1,0)\}$ are in fact a subset of B'', and therefore it is enough to show forbidden blocks of these shapes can be used to implement any recognizable grammar.

Let B be the set of shapes of tiles of G. We define the length of vector $v = (i, j)$ to be $l(v) = |i| + |j|$, and let m be the maximum length of a vector in the signal set of B. Define

$$M = [-m, m] \times [-m, m].$$

Then the set of states for the new grammar G' with neighborhood N' is $(\Sigma \cup \#)^M$, with the interpretation that if the state at position x on picture p is s, then $p[x+v] = s(v)$ for all $v \in M$. The state assignment rule is that the state holds the correct information at $v = (0,0)$.

Let

$$N = \{(0,1),(0,-1),(1,0),(-1,0)\}.$$

Define the right half H_r to be the set of positions $(i,j) \in M$ such that $j > 0$, and symmetrically define the left half H_l, the top half H_t and the bottom half H_b. If the state at the current position x is s, the local rule guarantees the following:

- If the right neighbor of x contains a $\#$, then

  $$\forall (i,j) \in H_r : s((i,j)) = \#,$$

  and symmetrically for the left neighbor, top neighbor and bottom neighbor.

- Otherwise, if the state of the right neighbor is s', then

  $$\forall v \in [-m,m] \times [-m+1,m] : s(v) = s'(v + (0,-1))$$

  and information is shared symmetrically with other neighbors.

We can now prove that if there exists a consistent assignment of states, then all information is correct in the states. Again, let p be the picture, x an arbitrary position on p and s the state of cell x. We proceed by induction on the distance of vectors $v \in M$ from the origin:

- The information in $s((0,0))$ must be correct because of the state assignment rule.

- Consider an arbitrary position v in s with distance $d > 0$ from the origin. Then there must be a neighbor $x + v'$ in some direction $v' \in N$ such that $l(v - v') < d$. If this neighbor is on the border of p, then the information in $s(v)$ must correctly hold $\#$, since $x + v$ must be outside the picture as well. Otherwise, let the state at $x + v'$ be s'. The information $s'(v - v')$ is correct by induction, and by the second local rule, $s(v) = s'(v - v')$, so also $s(v)$ must hold the correct information.

The new grammar G' is obtained from these rules, and the additional rule that a forbidden pattern of grammar G cannot appear in any of the states. It is clear that $\mathcal{L}(G) = \mathcal{L}(G')$.

As for the other direction, if a local grammar can be used to implement all the REC languages, then the signal set has to contain the natural basis as an easy corollary of the fact the finite languages $\{\begin{array}{|c|}\hline 0 \\ \hline 0 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline \end{array}\}$ and $\{\begin{array}{|c|c|}\hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline \end{array}\}$ are in REC. □

Corollary 2.20. *Every recognizable language is of both types*

$$\{\{(0,0),(0,1),(1,0),(1,1)\}\}$$

and

$$\{\{(0,0),(0,1)\},\{(0,0),(1,0)\}\}.$$

Tiles of these types are called square tiles *and* domino tiles*, respectively.*

Proof. Note that the neighborhoods defined by either type both contain the natural base of $\mathbb{Z}^2$. □

Note that within one neighborhood type, local languages can equally well be defined using sets of allowed tiles. When giving concrete tile sets, it is often more convenient to define languages this way. Building on the previous corollary and this remark, we define two other kinds of grammars for defining recognizable picture languages.

Definition 2.21. A *domino grammar* G is a pair (Σ, Δ, R, T), where T is a *set of allowed domino tiles*, and $R \subset \Sigma \times \Delta$ is the state assignment rule. The recognizable language $\mathcal{L}(G)$ is $\mathcal{L}(G')$ for the recognizable grammar $G' = (\Sigma, \Delta, R, T')$, where T' is the set of all domino tiles over Δ not in T. Similarly, we define *square grammars* using *sets of allowed square tiles*.

We also occasionally use domino grammars and square grammars with forbidden tiles, in which case this is explicitly mentioned.

We rarely give concrete tile sets, but instead, explain the local properties that are checked. These explanations will be directly translatable to tile sets, although a combinatorial explosion may be expected. Theorem 2.19 can then be used to construct a domino grammar or a square grammar, if one is desired. When we give concrete tilesets, we will usually draw the tiles instead of giving the corresponding functions in mathematical notation, and the relation R is explained in the text.

Finally, we give a few examples of languages in REC, many of which will be used later.

Example 2.22. *Every singleton language $\{p\}$ is in REC. A grammar for it is obtained by using a tile for each cell of p, and a local rule that arranges them into a rectangle in the correct way.*

Example 2.23. *The usual way to embed 1-dimensional languages into picture languages is to use subsets of*

$$L_{words} = \{p \mid \underline{p} = 1\}.$$

The recognizable languages of this form exactly correspond to the one dimensional regular languages. The language L_{words} itself is in REC by a grammar that forbids all vertical domino tiles of the form $\begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array}$ *where $a, b \in \Sigma$.*

Example 2.24. *Consider the tiling grammar given by the allowed tiles*

$$\{\begin{array}{|c|c|}\hline 0&0\\\hline 0&0\\\hline\end{array}, \begin{array}{|c|c|}\hline 1&0\\\hline 0&1\\\hline\end{array}, \begin{array}{|c|c|}\hline 0&1\\\hline 0&0\\\hline\end{array}, \begin{array}{|c|c|}\hline 0&0\\\hline 1&0\\\hline\end{array}\} \cup \{\begin{array}{|c|c|}\hline \#&\#\\\hline 1&0\\\hline\end{array}, \begin{array}{|c|c|}\hline \#&\#\\\hline 0&0\\\hline\end{array}, \begin{array}{|c|c|}\hline \#&1\\\hline \#&0\\\hline\end{array}, \begin{array}{|c|c|}\hline \#&0\\\hline \#&0\\\hline\end{array}\} \cup$$
$$\{\begin{array}{|c|c|}\hline \star&\star\\\hline \#&\#\\\hline\end{array}, \begin{array}{|c|c|}\hline \star&\#\\\hline \star&\#\\\hline\end{array}\} \cup \{\begin{array}{|c|c|}\hline \#&\#\\\hline \#&\#\\\hline\end{array}, \begin{array}{|c|c|}\hline \#&\#\\\hline \#&1\\\hline\end{array}, \begin{array}{|c|c|}\hline 1&\#\\\hline \#&\#\\\hline\end{array}, \begin{array}{|c|c|}\hline \#&\star\\\hline \#&\#\\\hline\end{array}, \begin{array}{|c|c|}\hline \#&\#\\\hline \star&\#\\\hline\end{array}\},$$

where $\star$ is a wildcard for $\{0, 1\}$, and R is obtained from the projection $f(0) = f(1) = a$. This gives us exactly the squares over a, because a diagonal line of ones can only start at the top left corner, and one must end at the bottom left corner.

2.2 Closure properties of REC

Describing recognizable picture languages directly using the definition by a grammar quickly becomes a tedious task. One simple way to deal with this problem is to build languages out of existing ones using closure properties of the class REC. In this section, we define two fundamental picture language operations, the concatenation of two pictures and the operation that iterates this. Both operations will be used a lot later on. We also define a few less orthodox operators as examples of what one can do with REC.

In most of the operations, the general idea is to splice one or more languages in REC together in different ways. This is done by 'drawing' a partition on the given picture, checking its consistency locally, and then recursively checking that the partition elements belong to the corrent languages, so to speak. In the tiling operation, this partition is arbitrary, whereas in the concatenation operation, the partition simply halves the picture.

We start with some auxiliary definitions. Since our pictures are actually just finite drawings on an infinite canvas, we can define a translation operation, which of course preserves the content of the picture (the matrix associated with the picture). The concept of translating was already explained in Section 2.1, we simply formalize this idea.

Definition 2.25. For any configuration $c' \in X^{\mathbb{Z}^2}$, for a finite set X, we denote by $\mathrm{T}_x c'$ the configuration c defined by $c[y] = c'[y - x]$. If c' is the configuration associated with picture p', this is the picture p' translated by the vector x.

Note that the translation by $(1, 0)$ moves the picture one step down, and the translation by $(0, 1)$ moves it one step to the right.

The following operations flip and rotate pictures. We give the operations for matrices first, and then define how they work on pictures.

Definition 2.26. Given a matrix M, we define the transpose of M by $M^T_{i,j} = M_{j,i}$, as usual. We also define the horizontal and vertical flips of $m \times n$ matrix M by $M^H_{i,j} = M_{i,n-j+1}$ and $M^V_{i,j} = M_{m-i+1,j}$, respectively. Finally, we define the clockwise rotation of M by $M^R_{i,j} = M_{m-j+1,i}$. Note that if M is an $m \times n$ matrix, M^T and M^R are $n \times m$ matrices, while M^H and M^V are both $m \times n$ matrices.

Definition 2.27. Let p be a picture and f a matrix operation. Then $f(p) = \mathrm{T}_{\mathrm{offset}(p)} f(\mathrm{cont}(p))$.

Note that with this definition, matrix operations commute with translations. Usually, our language operations have this property. However, even if the operations were defined as rotating around the origin and flipping over the axes, being closed under these operations would of course be equivalent to being closed under the translation-commuting versions due to the fact languages are closed under translations.

The first 'splicings' we introduce are ones that simply glue pictures together, edge-to-edge.

Definition 2.28. Let p_1 and p_2 be two pictures in normal form. If $|p_1| = |p_2|$, we will denote by $\boxed{\begin{array}{c} p_1 \\ \hline p_2 \end{array}}$ or $p_1 \ominus p_2$ the picture p defined by

$$p[x] = \begin{cases} p_1[x] & \text{if } x \in \mathrm{dom}(p_1) \\ \mathrm{T}_{(\underline{p_1},0)} p_2[x] & \textit{otherwise} \end{cases}$$

The alphabet of p is the union of the alphabets of p_1 and p_2. Similarly, assume p_1 and p_2 are in normal form. Then we denote by $\boxed{p_1 \mid p_2}$ or $p_1 \oplus p_2$ the picture p defined by

$$p[x] = \begin{cases} p_1[x] & \text{if } x \in \mathrm{dom}(p_1) \\ \mathrm{T}_{(0,|p_1|)} p_2[x] & \textit{otherwise} \end{cases}$$

Clearly $\boxed{p_1 \mid p_2} = \boxed{\begin{array}{c} p_1^T \\ \hline p_2^T \end{array}}^T$. Informally, we will refer to these operations as gluing.

Of course, we may apply the previous definition to pictures not in normal form by first translating them to this form.

An operation on pictures is lifted into an operation on languages in the way it is usually done in set theory. For unary functions, we already defined in Definition 2.9 how a scalar function behaves on a set. The following generalizes this idea.

Definition 2.29. Given two picture languages L_1 and L_2, we denote by $\boxed{L_1 \mid L_2}$ and $\boxed{\begin{array}{c} L_1 \\ \hline L_2 \end{array}}$ the sets of legal gluings of pictures in these classes. Similarly, for any other operation on pictures, there is a corresponding operation on languages with the same notation.

Note that our operations are often partial functions, so we only take all the legal combinations of arguments. So for instance, even if for some operation $+$, $p_1 + p_2$ is not defined, $\{p_1\} + \{p_2\}$ will be the empty language instead of an invalid application.

Gluing can be iterated in a natural way, corresponding to the star operation of one-dimensional regular languages.

Definition 2.30. If $L \subset \Sigma_*^*$, we write $L^{n\oplus}$ for the language $\{p_1 \oplus \cdots \oplus p_n \mid \forall i : p_i \in L\}$. Similarly, we write $L^{n\ominus}$ for $\{p_1 \ominus \cdots \ominus p_n \mid \forall i : p_i \in L\}$. We write $L^{*\oplus} = \bigcup_n L^{n\oplus}$ and $L^{*\ominus} = \bigcup_n L^{n\ominus}$ for the closures under these operations. These operations are referred to as the 'star operations'.

The previous operation is not the lifting of a scalar operation to an operation on languages, unless the operation takes a set of pictures as its argument. Of course, we could define a similar operation $p \mapsto p^n$ for single pictures, and obtain a language operation from it. However, it is not very hard to see, once we have the proper tools, that REC is not closed under the operation mapping $L \to \{p^{n\oplus} \mid p \in L\}$ if $n \geq 2$.

Theorem 2.31. *The class REC is closed under transposing, and horizontal and vertical flipping.*

Proof. Even LOC is closed under these operations by applying the respective operations on the set of forbidden patterns, and the operations commute with symbol-to-symbol mappings. The is, let L be a language in REC, and let $L = f(L')$ for some local language L'. Then, if g is one of the operations listed, $g(L) = g(f(L')) = f(g(L'))$, where $g(L')$ is a local language, and thus $g(L)$ is in REC. □

The following theorem is straightforward to prove.

Theorem 2.32. *REC is closed under monotone boolean functions (union and intersection and their compositions).*

Theorem 2.33. *The class REC is closed under gluing. That is, given* $L_1, L_2 \in$ *REC, the languages* $\boxed{\begin{array}{|c|}\hline L_1 \\ \hline L_2 \\ \hline \end{array}}$ *and* $\boxed{\begin{array}{|c|c|}\hline L_1 & L_2 \\ \hline \end{array}}$ *are in REC.*

Proof. By symmetry (and by $\boxed{\begin{array}{|c|c|}\hline p_1 & p_2 \\ \hline \end{array}} = \boxed{\begin{array}{|c|}\hline p_1^T \\ \hline p_2^T \\ \hline \end{array}}^T$), it is enough to show REC is closed under vertical gluing. Let $L_1, L_2 \in$ REC with the domino grammars $(\Sigma, \Delta_1, R_1, T_1)$ and $(\Sigma, \Delta_2, R_2, T_2)$, respectively. We may assume $\Delta_1 \cap \Delta_2 = \varnothing$. We define a new domino grammar (Σ, Δ, R, T) by

$$\Delta = \Delta_1 \cup \Delta_2,$$

$$R = R_1 \cup R_2$$

$$T = T_1 \cup T_2 \cup T_t$$

where T_t is the set of transition tiles

$$T_t = \boxed{\begin{array}{|c|}\hline \Delta_1' \\ \hline \Delta_2' \\ \hline \end{array}},$$

where Δ_1' and Δ_2' are the border letters $\Delta_1' = \{a \in \Delta_1 \mid \boxed{\begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array}} \in T_1\}$ and $\Delta_2' = \{a \in \Delta_2 \mid \boxed{\begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array}} \in T_2\}$. Then it is clear that pictures are either from L_1 or L_2, or have a picture of L_1 on the first k rows for some k, and a picture of L_2 on the lines below it. Since the empty picture is in all REC languages, this is exactly the set of pictures of $\boxed{\begin{array}{|c|}\hline L_1 \\ \hline L_2 \\ \hline \end{array}}$. □

Theorem 2.34. *REC is closed under the star operations.*

Proof. Let $L \in$ REC. Again, it is enough to show that $L^{*\ominus}$ is in REC. Let $(\Sigma, \Delta_1, R_1, T_1)$ and $(\Sigma, \Delta_2, R_2, T_2)$ be two domino grammars for L with disjoint state sets (by renaming letters). Let Δ, R and T be as in the previous proof, except using the transition tiles

$$T_t = \boxed{\begin{array}{|c|}\hline \Delta_1^b \\ \hline \Delta_2^t \\ \hline \end{array}} \cup \boxed{\begin{array}{|c|}\hline \Delta_2^b \\ \hline \Delta_1^t \\ \hline \end{array}}$$

instead, where

$$\forall i : \Delta_i^b = \{a \in \Delta_i \mid \boxed{\begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array}} \in T_i\}$$

$$\forall i : \Delta_i^t = \{a \in \Delta_i \mid \boxed{\begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array}} \in T_i\}$$

□

Definition 2.35. The cartesian product of the languages L_1 and L_2 is the set

$$L_1 \times L_2 = \{p \in (\mathrm{alph}(L_1) \times \mathrm{alph}(L_2))^*_* \mid \pi_i(p) \in L_i\}$$

The following should be obvious.

Theorem 2.36. *REC is closed under cartesian products.*

Note that the cartesian product of two languages is strictly speaking almost never their set theoretic cartesian product. However, if we project the pictures in this language to their L_1 and L_2 sides, we get a set theoretic relation between the two languages with the additional constraint that if two pictures are associated with each other, they have to have the same size.

Before defining more complicated operations REC is closed under, we introduce another 'programming technique'. Instead of building languages out of existing ones, this technique deals more directly with the specifics of REC. What we do is logically separate accepting a picture language into phases that each add certain information to the symbols, and finally we check the inclusion of the picture with added information in another language, which is hopefully easier to show recognizable. This information may be added nondeterministically.

Definition 2.37. Let $L \subset L_1 \times L_2$. Then we say an REC grammar G *implements* L *as a relation* if $\mathcal{L}(G) = L$.

Lemma 2.38. *For $1 \le i \le n$, let L_i be languages, let L_n be recognizable, and let*

$$R_i \subset \mathit{alph}(L_i)^*_* \times \mathit{alph}(L_{i+1})^*_*$$

be REC implementable relations. Further assume

$$p \in L_1 \iff \exists q \in L_n : (p, q) \in R_1 \circ \cdots \circ R_{n-1}.$$

Then L_1 is recognizable. The type of the grammar obtained for L_1 is the union of the types of grammars of the R_i and L_n.

Proof. Let the grammars $G_i = ((\Sigma_i \times \Sigma_{i+1}), \Delta_i, C_i, T_i)$ implement R_i for all i, and let $G' = (\Sigma_n, \Delta', C', T')$ implement L_n. We construct a grammar G with $\mathcal{L}(G) = L_1$.

Let $G = (\Sigma_1, \Delta, C, T)$ where

$$\Delta = (\prod_{1 \le i \le n} \Sigma_i) \times (\prod_{1 \le i \le n-1} \Delta_i) \times \Delta',$$

and let (π_{Σ_i}), (π_{Δ_i}) and $\pi_{\Delta'}$ be the corresponding projections.

The state assignment rule of G can put state s on top of $a \in \Sigma_1$ if

- $\pi_{\Sigma_1}(s) = a$.
- for all $1 \le i \le n-1$, $\pi_{\Delta_i}(s)$ is allowed on top of $(\pi_{\Sigma_i}(s), \pi_{\Sigma_{i+1}}(s))$ in G_i.
- $\pi_{\Delta'}(s)$ is allowed on top of $\pi_{\Sigma_n}(s)$ in G'.

As for the local rule, for each forbidden tile T in any of the grammars $H \in \{G_1, \ldots, G_{n-1}, G'\}$ with states $S \in \{\Delta_1, \ldots, \Delta_{n-1}, \Delta'\}$, every tile U of the same shape as T with $\pi_S(U) = T$ is forbidden.

Let us show the language of G is exactly L_1: First, let $p \in L_1$. Then, there exists $q \in L_n$ such that $(p, q) \in R_1 \circ \cdots \circ R_{n-1}$, that is,

$$p = p_1 R_1 p_2 R_2 \cdots R_{n-1} p_n = q.$$

For $1 \leq i \leq n-1$, let $r_i \in (\Delta_i)^*_*$ be assignments of states proving $p_i R_i p_{i+1}$, and let $r \in \Delta'$ be an assignment of states proving $q \in L_n$. then, by letting $\pi_{\Sigma_i}(p) = p_i$, $\pi_{\Delta_i}(p) = r_i$ and $\pi_{\Delta'}(p) = q$, we obtain a proof that $p \in \mathcal{L}(G)$, since the local rule checks exactly the same things as the individual grammars.

The converse claim follows similarly, since from an assignment of states proving $p \in \mathcal{L}(G)$, we obtain pictures p_i and proofs r_i as in the proof of the first direction, giving a picture $q \in L_n$ such that

$$p = p_1 R_1 p_2 R_2 \cdots R_{n-1} p_n = q,$$

in which case $p \in L_1$ by the assumption of the theorem. □

Note that by the fact that picture classes are closed under bijective alphabet substitutions, for every picture class CLS, $L \in \text{CLS} \iff \{p \times p \mid p \in L\} \in \text{CLS}$ holds. Similarly, if $R \subset L_1 \times L_2$ is a recognizable relation, also $L'_1 \times L_2$ is recognizable for $L'_1 = \{p \times p \mid p \in L_1\}$. Therefore, whenever we can compute a relation, we can also add the information computed by the relation on top of the existing picture. Such relations are called *incremental relations*.

As an example of why the previous lemma is useful, consider the problem of comparing the first and the last columns of pictures. While it is simple enough to send a signal from left to right on each row, if we later wanted to compare the first column to the second to last column instead, we could not directly use the first result in any way.

A nicer way is to implement a relation that adds to the symbol at each position the symbols of the neighboring cells, cycling information over the borders. We can then apply this relation once if we want to compare the first row to the last one, and twice if we want to compare it to the second to last row.

Also the relation adding neighbor information without wrapping around the borders is useful, and we will need it later. This is called the *power signal construction*, and can directly be extracted from the proof of Theorem 2.19.

Lemma 2.39. *Let Σ and m be arbitrary, and let $M = [-m, m] \times [-m, m]$. Then the relation containing pairs $(p, p') \in \Sigma^*_* \times ((\Sigma \cup \{\#\})^M)^*_*$ such that*

$$\forall x \in dom(p) : \forall v \in M : p[x + v] = p'[x](v)$$

is recognizable.

As an application of the lemma, we show an interesting closure property of REC. This is the tiling operation, which paints a new picture by putting pictures of another language together in an arbitrary way.

Definition 2.40. Let L be a picture language. We define the *tilings by L* as

$$L^\star = \{p \in \text{alph}(L)^*_* \mid \exists p_1, \ldots, p_n \in L : \forall y \in \text{dom}(p) : \\ (\exists! i : p[y] = p_i[y] \wedge \forall j \neq i : p_j[y] = \#)\}.$$

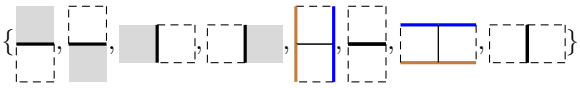
That is, a picture is in $L^\star$ if it can be partitioned into pictures of L. Note that the pictures p_i in the definition need not be in normal form.

Theorem 2.41. *REC is closed under the tiling operation.*

Proof. Let L be the language of a domino grammar G with *forbidden tiles*. We implement the language $L^\star$ as an REC grammar, using the previous lemma to split the problem into two parts. First, we implement a relation that guesses a partition of the picture into rectangles by adding borders to cells. After this, all we have to do is guess states of L for each position of the picture, and consider the borders drawn by the relation as virtual borders from the perspective of the local rule of L.

The relation that adds the borders is from pictures over Σ to pictures over $\Sigma \times \{0,1\}^4$, where the part $\{0,1\}^4$ gives the sides of cells that have a border, 1 meaning there is a border, 0 meaning there is none, for the north, east, south and west sides, respectively. We think of the borders as being between the cells rather than on their sides.

It is enough to show that partitions into rectangles form a language L' in REC, since then the relation adding such borders is just $\Sigma_\star^* \times L'$, and REC is closed under cartesian products. It turns out that the language L' is even local, and given by the following set of allowed domino tiles:

$T =$ {  }

A thick line represents a 1, and a thin line represents a 0. Dashed lines are wildcards, and can be filled arbitrarily, and brown and blue lines range over $\{0,1\}$. Borders occur in pairs. For instance, whenever a cell has a right wall, its east neighbor will have a left border, implementing the idea that a border is actually between the cells.

First note that any partition is clearly allowed by these tiles, that is, given a partition of a large rectangle into smaller rectangles, the borders of the smaller rectangles must satisfy the local constraints: The first four tiles make sure that at least one small rectangle is next to each border of the large rectangle, and the four other tiles make sure that a new small rectangle must start where another one ends, and that borders must continue unless they touch the middle of a perpendicular border. All of these properties are clearly satisfied by rectangular partitions.

We still have to show that these local rules only allow rectangular partitions. Therefore, consider an arbitrary picture p in this language. It is enough to show that each cell is contained in a unique rectangle drawn by a border. Let y be an arbitary position on p, and let R be the set of positions reachable from y without crossing a border.

We first note that if there are two cells a and b on the same row, with a vertical border v somewhere between them on the same row, these two cells can never be connected: The vertical bar v always has a continuation either (say) south or east on one side, and either (say) north and west on one side, until it reaches the border of the large rectangle. This is true since at all junctions, a border is either continued, or it branches in at least two directions. Any path between a and b would clearly have to cross one of these paths. See Figure 2 for an illustration of this.

Now it is clear that R cannot enclose another connected region from all sides, and that there can be no borders between two adjacent interior cells of R. This

Figure 2: Any path between a and b would have to intersect the continuation of v.

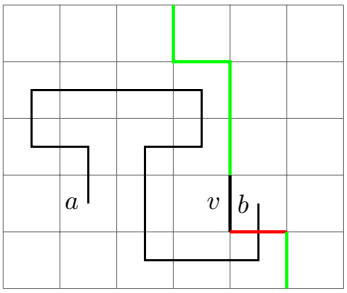
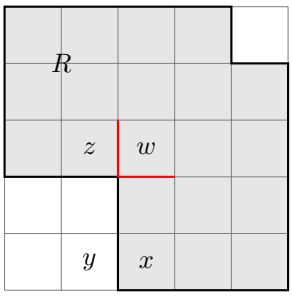


Figure 3: If R is not a rectangle, it can not be connected.



means R is hv-convex, that is, closed under adding cells between two cells on the same row or column. We will prove that R must in fact be a rectangle. So let us assume, on the contrary, that R is not a rectangle. Then, by following its border, we necessarily find an 'outward turn', which gives a pair of cells in R that cannot be connected.

Let us make this slightly more precise. By the 8-fold symmetry of the situation, we can assume there is a cell x on the last row of R such that the west neighbor y of x is not in R, but there is a cell in R on the column of y. Consider the first cell z of R on the column of y, going upward. Then because of connectedness and hv-convexity, the east neighbor w of z is in R. But then w has either a west border or a south border due to the local rules. Both would disprove connectedness of R: either w is not connected to x or it is not connected to z. This is a contradiction, so R must be a rectangle. See Figure 3 for an illustration of this.

Now, we still have to show how to check that rectangular partitions with pictures of L in the small rectangles form a recognizable language. This is much simpler, and the tiles are obtained using a similar idea as the one used in the proof of Theorem 2.34. The states allowed on a given symbol of $\Sigma \times \{0,1\}^4$ in the new grammar are the same as the ones allowed on the Σ part in the original grammar. We modify the horizontal tiles to use the borders drawn on the picture as virtual borders for the small pictures:

- For each forbidden tile $\boxed{\text{a}\,|\,\text{b}}$, where $a, b \in \Sigma$, in the local domino gram-

mar of G, we forbid the tiles $\boxed{a}\boxed{b}$.

- For each forbidden tile $\boxed{\#}\boxed{\text{b}}$, where $b \in \Sigma$, we forbid the tiles $\boxed{a}\boxed{b}$ for all $a \in \Sigma$.
- For each forbidden tile $\boxed{\text{a}}\boxed{\#}$, where $a \in \Sigma$, we forbid the tiles $\boxed{a}\boxed{b}$ for all $b \in \Sigma$.

We obtain the forbidden vertical tiles symmetrically.

Then cells are considered adjacent in the small pictures if there is no border between them, and otherwise they are seen as being next to the border of the small picture, so it is clear that the small pictures will contain pictures in L. □

The proof also gives another interesting example of a language in LOC (and in particular in REC). A formal definition for the language could be extracted from the definition of the tiling operation by taking the language of pictures over $\Sigma \times \{0,1\}^4$, with borders drawn around them as is done by L'.

Example 2.42. *The language of rectangular partitions (drawn as side borders on the cells) is local.*

We give two more examples of closure properties. Both are used in the proof of Theorem 3.18.

Theorem 2.43. *REC is closed under removing the top rows of pictures.*

Proof. Given L in REC with domino grammar G, we need to construct a recognizable grammar for the language $\{p[[2,\overline{p}] \times [1,|p|]] \mid p \in L\}$. The states of the new language are the same as before, except that the states at the top row also encode the hypothetical contents and states of the removed row. Of course, the state alphabet is the same on all rows, so we then have to store two symbols information on all rows. We ignore this information on all rows except the top row.

The local rule works as the one in G, except for the top row, where the horizontal rule of G is checked on both the actual top row and the hypothetical top row of the picture whose top row was deleted. The vertical rule checks that the hypothetical top row could be a top row, and that the actual top row could have the hypothetical one on top of it according to the local rule of G. □

Corollary 2.44. *REC is closed under removing the top and bottom rows of pictures, and the leftmost and rightmost columns of pictures.*

Proof. The claims can easily be reduced to Theorem 2.43 by using suitable rotations. □

Theorem 2.45. *REC is closed under removing the odd rows of pictures (counting top-down).*

Proof. Given $L \subset \Sigma_*^*$, we construct a domino grammar $G' = (\Sigma, \Delta', R', T')$ for showing that the language containing the odd rows of even height pictures of L is in REC. Then, since L_{words} is in REC, and REC is closed under vertical gluing, we obtain, using the same grammar, that the odd rows of odd height pictures is in REC. Since REC is also closed under union, the claim follows.

Let $G = (\Sigma, \Delta, R, T)$ be a domino grammar for L. Then, we take $\Delta' \subset \Delta \times \Sigma \times \Delta$ such that

$$\forall (s,b,s') \in \Delta' : \boxed{\begin{array}{c} s \\ \hline s' \end{array}} \in T$$

as the set of states. The state assignment rule is

$$(a,(s,b,s')) \in R' \iff (a,s) \in R \wedge (b,s') \in R.$$

The horizontal domino tiles of Q' are

$$\boxed{(s_1,b_1,s_1') \mid (s_2,b_2,s_2')} \in T' \iff \boxed{s_1 \mid s_2} \in T \wedge \boxed{s_1' \mid s_2'} \in T,$$

where $\#$ is thought of as $(\#,\#,\#)$. The vertical domino tiles are

$$\boxed{\begin{array}{c} (s_1,b_1,s_1') \\ \hline (s_2,b_2,s_2') \end{array}} \in T' \iff \boxed{\begin{array}{c} s_1' \\ \hline s_2 \end{array}} \in T,$$

where $\#$ is thought of as $(\#,\#,\#)$.

It is clear that if the Σ layer of the states, for each row r, were added as a row under r, we would obtain an even height picture in L, and vice versa. □

Corollary 2.46. *REC is closed under removing the odd or even rows or columns counted top-down or bottom-up.*

Proof. All of these claims can easily be reduced to removing the odd rows by using suitable rotations, and additions and removals of rows. □

Before concluding, let us briefly discuss the implications of picture classes being closed under bijective symbol-to-symbol projections. As we already mentioned in Section 2.1, REC is closed under symbol-to-symbol projections. Consider the following 'weaker' projection operation.

Definition 2.47. Let $\Sigma = \Sigma_1 \times \Sigma_2$, and let $L \in \Sigma_*^*$. Then we define the *projection of L on the first coordinate* as $\{p \in (\Sigma_1)_*^* \mid \exists p' \in (\Sigma_2)_*^* : p \times p' \in L\}$.

REC is clearly closed under such an operation: Given a picture p over Σ_1, we can add an arbitrary picture over Σ_2 on top of p using a recognizable relation. The claim then follows from the fact that L is recognizable by the assumption. In fact, due to the closure property all classes have, we can prove a kind of converse to this. Consider the following temporary picture class.

Definition 2.48. LOCP is the smallest class containing LOCP that is closed under projections on the first coordinate.

Example 2.49. *LOCP = REC.*

Proof. Clearly LOCP $\subset$ REC. For the other direction, we simply need to show LOCP is closed under all symbol-to-symbol projections.

Consider a language $L \in$ LOCP over some alphabet Σ, and let $f : \Sigma \to \Sigma'$ be an arbitrary surjection. Consider the injection $g : \Sigma \to (\Sigma' \times \Sigma)$ defined by $g(a) = (f(a), a)$. It is a bijection onto its image $S = g(\Sigma)$, and therefore $g(L) \subset S_*^*$ is in LOCP, since picture classes are closed under symbol-to-symbol bijections. But the projection of $g(L)$ onto the first coordinate is just $f(L)$, which proves the claim. □

In general, when an operation is given that assumes a structure for the alphabet, we may actually think of the operation as first adding some structure on the alphabets, and then using the operation for this structure.

For example, we might define an operation that takes pictures over groups as inputs, and produces pictures where, at each cell, the symbol has been multiplied by its right neighbor, wrapping over borders, and then squared. The corresponding language operation would allow much more freedom, since any alphabet with the same or smaller cardinality as group G can be thought of as G by taking a larger alphabet, and a bijection with G.

As a simpler example, operations such as projection on the first coordinate that work on languages over cartesian products can be thought of as working on decompositions of the the alphabet.

2.3 REC on general graphs, and languages not in REC

Tiling systems are used to define natural language classes on many kinds of 'structures'. In addition to word languages and picture languages, also languages over trees, acyclic graphs and infinite words have a rich theory, and tiling system based language families are studied the most in each case. All of these are special cases of graphs. In the following, we define the classes $\mathcal{G}$-REC for languages over an arbitrary graph family $\mathcal{G}$, and give a lemma for showing a language is not in $\mathcal{G}$-REC, for a given $\mathcal{G}$. We show two ways of using the lemma in the case of picture languages.

We consider REC on general graphs for two reasons. First of all, we will define multiple ways of representing pictures as graphs in later chapters, so it makes sense to define the *canonical* graph representation of pictures, and show REC on pictures corresponds to the recognizable graph languages, when this graph representation is used for the pictures. The third reason is that in Section 2.4, the canonical graph representation of a picture directly gives its logical structure as well.

Our graphs often have a quite a bit of additional structure. We give a name to such graphs to avoid listing the many properties at every turn.

Definition 2.50. A *canvas*, is a directed, rooted, 'road-colored multigraph' (rather a graph with a multi-road-coloring) with self-loops, which can be either vertex-labeled or not such that there is a path to every node from the root. That is, a labeled canvas is a 7-tuple $(V, E, r, \Sigma, \Gamma, l, f)$ where

- V is the set of *vertices*
- $E \subset \{(x, y) \mid x, y \in V\}$ is the set of *edges*
- $r \in V$ is the *root*
- Σ is the *alphabet*
- Γ is the set of *directions*
- $l : V \to \Sigma$ is the *vertex labeling*
- $f : E \to 2^{\Gamma}$ is the *road-coloring*

and the road-coloring f has following property

$$\forall x \in V : \forall g \in \Gamma : \exists! y \in V : (x, y) \in E \wedge g \in f((x, y)).$$

Given such x and g, we write xg for the unique y such that $g \in f((x, y))$. The last requirement is that

$$\forall v \in V : \exists u \in \Gamma^* : v = ru.$$

An unlabeled canvas is defined similarly, dropping Σ and l in the definition.

When there is no danger of confusion, we will also refer to canvases as just graphs. Encodings of pictures as graphs are always canvases.

Note that any node v of any canvas can be written as at least one element of Γ^* representing a path from the root to v. Conversely, every element of Γ^* corresponds to a unique vertex of the canvas.

Definition 2.51. A canvas family $\mathcal{G}$ is a set of nonlabeled canvases all using the same direction alphabet Γ. For each canvas family $\mathcal{G}$ there is a corresponding family $\mathcal{L}(\mathcal{G})$ of languages. Such languages are subsets of vertex-labeled versions of canvases in $\mathcal{G}$ all labeled over the same (but arbitrary) alphabet Σ.

The definitions of LOC and REC have straightforward generalizations to all canvases.

Definition 2.52. A Γ-block is a finite subset of Γ^*. A Γ-tile over Σ is a function from a Γ-block to Σ.

Similarly to the case of the discrete plane, we may talk about the occurrences of a Γ-tile on a labeled canvas. We say that the tile t occurs on the labeled canvas p at node v if

$$\forall x \in \mathrm{dom}(t) : l(vx) = t(x).$$

Definition 2.53. The $\mathcal{G}$-LOC languages are the subset of $\mathcal{L}(\mathcal{G})$ of languages $L \in \mathcal{L}(\mathcal{G})$ for which there exists a finite set of tiles T such that

$$p \in L \iff \text{none of the tiles in } T \text{ occur in } p.$$

Again, $\mathcal{G}$-REC is defined by taking the symbol-to-symbol projections of languages in $\mathcal{G}$-LOC.

Sets of domino tiles can be proven universal for all canvas families. We skip the proof, which is essentially the same as in the case of picture languages.

Theorem 2.54. *For every $\mathcal{G}$-REC language L, there exists an underlying local grammar defining it that only uses tiles of shapes $\{\epsilon, g\}$ for $g \in \Gamma$.*

The theorem guarantees the existence of *domino grammars* for all $\mathcal{G}$-REC languages, that is, grammars (Σ, Δ, R, T) with the same interpretation as in the REC case, such that T is a set of tiles of shapes $\{\epsilon, g\}$ for $g \in \Gamma$.

Let us now return to pictures. In order to find a natural specialization for pictures of the general definition of $\mathcal{G}$-REC, we have to decide what the 'canonical representation' of a picture as a canvas should be. Two ways to do this are given in the following, one that only looks at the underlying matrix and one

that better corresponds to our definition of a picture as an infinite configuration. In the first representation, the canvas corresponding to a picture has edges between adjacent cells of the picture, and an inescapable border around it. In the second one, all pictures correspond to infinite graphs (with $\mathbb{Z}^2$ as the set of vertices).

Definition 2.55. Given $p \in \Sigma_*^*$, the *bordered representation* of p is the canvas $(V, E, r, \Sigma, \Gamma, l, f)$, where

$$V = [0, \underline{p} + 1] \times [0, |p| + 1]$$

$$E = E_{up} \cup E_{right} \cup E_{down} \cup E_{left}$$

$$l(x) = \begin{cases} p[x] & \text{if } x \in \mathrm{dom}(p) \\ \# & \text{otherwise} \end{cases}$$

$$\Gamma = \{\vartriangle, \triangleright, \triangledown, \triangleleft\}$$

$$f(e) \ni \begin{cases} \vartriangle & \text{if } e \in E_{up} \\ \triangleright & \text{if } e \in E_{right} \\ \triangledown & \text{if } e \in E_{down} \\ \triangleleft & \text{if } e \in E_{left} \end{cases}$$

$$r = (1, 1)$$

where

$$E_{right} = \{(x, x + (0,1)) \mid x \in [0, \underline{p} + 1] \times [0, |p|]\} \cup \{(x, x) \mid x \in [0, \underline{p} + 1] \times \{|p| + 1]\}\}$$

and E_{up}, E_{down} and E_{left} are defined similarly, in the obvious way.

Definition 2.56. Given $p \in \Sigma_*^*$, the *unbordered representation* of p is the unlabeled canvas $(V, E, r, \Sigma, \Gamma, l, f)$, where $V = \mathbb{Z}^2$, and E, r, l and f and r are as in the previous definition, except that the components of E are defined differently:

$$E_{right} = \{(x, x + (0,1)) \mid x \in \mathbb{Z}^2\}$$

and E_{left}, E_{up} and E_{down} are defined similarly, in the obvious way.

It is clear that the bordered representations of pictures of REC languages correspond exactly to the $\mathcal{G}$-REC languages of pictures bordered with #, where $\mathcal{G}$ is the set of underlying canvases of pictures. The intersections of $\mathbb{Z}^2$-REC languages with the language of unbordered representations of pictures, however, is a much larger class than the unbordered representations of REC languages, and we will not discuss it further.

We now prove a generalized version of the *exchange lemma*, often called Matz' lemma in the case of picture languages [13], which is the usual tool for showing a language is not in REC. We apply Matz lemma as it is commonly used to find examples of languages outside REC. At least [13] and [18] have asked if this usual way of dividing a picture in two and swapping the sides can always be used to sho a language is outside of REC. We mention a language for which this is not the case, and use the lemma in a different way to show it is outside REC.

An unordered path in a graph G is a path in the undirected version G° of G defined as

$$G^\circ = (V, E \cup E^R).$$

We define the undirected version of a canvas as the undirected version of its underlying graph, and we obtain the undirected paths in a canvas the same way as for graphs. Note that for canvas G, G° is always a connected graph.

Definition 2.57. Given a canvas $G = (V, E, r, \Sigma, \Gamma, l, f)$, a cut C of G is a pair (V_1, E', V_2) where E' is a subset of E such that $G^\circ - (E' \cup E'^R)$ has at least two components, and V_1 and V_2 are the partition of V that E' induces in G°. The *size* of a cut (V_1, E', V_2) is $|E'|$.

Definition 2.58. Let G be a canvas and let p_1 and p_2 be two labeled canvases with underlying canvas G. Let $C = (V_1, E', V_2)$ be a cut of G. Then we denote by $\mathrm{swap}(p_1, C, p_2)$ the labeled canvas on G defined by the the vertex labeling function l such that

$$l(v) = \begin{cases} p_1[v] & \text{if } v \in V_1 \\ p_2[v] & \text{if } v \in V_2 \end{cases}$$

Lemma 2.59. *Let L be the language of a domino grammar with m states, let $p_1, \ldots, p_n \in L$ be labeled canvases with the same underlying unlabeled canvas G, and let $C = (V_1, E', V_2)$ be a cut of G of size k. Then if $n > (m^2)^k$, we have*

$$\exists j \neq j' : \mathit{swap}(p_j, C, p_{j'}) \in L \wedge \mathit{swap}(p_{j'}, C, p_j) \in L$$

Proof. Let Δ be the set of states used by the grammar, let X be the local grammar of the $\mathcal{G}$-REC grammar corresponding to L, and let L' be the language of X. Since $\forall i : p_i \in L$, there are pictures $q_i \in L'$ obtained by adding a Δ layer on each of the pictures p_i. Let $S = \{v \mid \exists x : (v, x) \in E' \vee (x, v) \in E'\}$. Then $|S| \leq 2|E'|$. Let $s(q_i)$ be the restriction of q_i to S considered as a function in Δ^S. Since $|S| \leq 2k$, there are at most $m^{2k} = (m^2)^k$ such functions, and therefore there must be a repetition in the values $s(q_i)$ in the $n > (m^2)^k$ values of i.

Let $j \neq j'$ be two indices such that $s(q_j) = s(q_{j'})$, and consider $q = \mathrm{swap}(q_j, C, q_{j'})$, which is well-defined since the q_i all have the same underlying canvas G. This picture is in L': If it were not, then there would be a forbidden tile around some vertex v. But if $v \in V_1$, then $\forall g \in \Gamma$, the label of vg in q is the same as its label in q_j: Any node whose label is taken from $q_{j'}$ will either be in V_1 or in S. In the first case, its label is the same as in q_j by the definition of q. In the second case, $vg \in S$ and $s(q_j) = s(q_{j'})$ imply $q[vg] = q_2[vg] = q_1[vg]$. If $v \in V_2$, the claim is symmetric.

This implies the new picture q satisfies the local constraints of X. Therefore, if we also swap the pictures p_j and $p_{j'}$ using the cut C, we note that q satisfies the state assignment rule and the local rule, and thus constitutes proof that $\mathrm{swap}(p_{j'}, C, p_j)$ is in L. □

The lemma can be used to show a language cannot be implemented using a local grammar with m states. Therefore, to prove a language is not in $\mathcal{G}$-REC using any local grammar, we need to find, for *every* m, a proof that m is not enough to implement the whole language. Roughly speaking, this means that the amount of information transmitted over some cuts grows superexponentially with the size of the cuts. As an application of the lemma, we find a simple first example of a language outside REC.

Example 2.60. *If $|\Sigma| > 1$, the language*

$$L_{c\neq c} = \{p \in \Sigma_*^* \mid \nexists\, i \neq j : p[*, i] = p[*, j]\}$$

is not recognizable.

Proof. We may assume $|\Sigma| = 2$. For any $h > 0$, consider the set C of all columns of height h over Σ. Then $|C| = 2^h$. Therefore, there are $\binom{2^h}{2^{h-1}}$ ways to split C into two sets of equal size. Now, for each such a split

$$H_1 \cup H_2 = C, H_1 \cap H_2 = \varnothing,$$

take one of the pictures $p = p_1 \oplus p_2$, where the set of columns in p_i is H_i, as a representative, to obtain $\binom{2^h}{2^{h-1}}$ pictures of $L_{c\neq c}$ of height h. Let the number of states for the local domino grammar be $|\Delta| = m$. Then, to fill the assumptions of the lemma, take h big enough that

$$\binom{2^h}{2^{h-1}} > (m^2)^h \text{ (see Appendix A).}$$

The edges between p_1 an p_2 give a cut for the pictures obtained this way, so by Lemma 2.59 we may swap two pictures with respect to this cut to obtain a new picture in L. But it is clear that the two sides of any picture obtained this way will have a nontrivial intersection of columns, and thus cannot be in $L_{c\neq c}$. □

From the previous language we also get a nonclosure property for REC:

Theorem 2.61. *REC is not closed under complementation.*

Proof. Consider the language $L_{c=c} = \{p \in \Sigma_*^* \mid \exists i \neq j : p[*, i] = p[*, j]\}$, which is clearly the complement of language $L_{c\neq c}$. It is enough to prove this language is in REC.

Let $a \in \Sigma$. Then $L_{a?a} = a\Sigma^* a$ is in REC by a domino grammar that checks that pictures have height 1 and that the corner symbols are a's. Then also $L_{x?x} = \bigcup_{a\in\Sigma} L_{a?a}$ is in REC, since REC is closed under monotone boolean operations. REC is also closed under the vertical star operation, so $L_{l=r} = \{p \in \Sigma_*^* \mid p[*, 1] = p[*, |p|]\} = L_{x?x}^{*\ominus}$ is in REC. Finally, since Σ_*^* is in REC, and REC is closed under horizontal gluing, we get that $L_{c=c} = \Sigma_*^* \oplus L_{l=r} \oplus \Sigma_*^*$ is in REC. □

The acyclic graphs are not recognizable either, at least using most natural visual representations, assuming edges are able to cross. A proof of this is given in at least [7] and [12] in the context of comparing REC with the so-called alternating finite automata, which we will consider later. We choose the following representation for graphs.

Definition 2.62. Consider the alphabet $\Sigma = \{f \mid f : \Gamma \to 2^\Gamma\}$. To each picture $p \in \Sigma_*^*$, we associate the directed graph (V, E), where

$$V = \mathrm{dom}(p) \times \Gamma,$$

and

$$E = \{((x, g), (xg', g'^{-1})) \mid g' \in p[x](g)\}.$$

The language Σ_*^* is called L_{graphs}. The subset of L_{graphs} such that the graphs corresponding to the pictures do not have cycles is called $L_{acyclic}$.

Here, giving the name L_{graphs} to $\Sigma_\star^*$ has the sole purpose of giving meaning to the intuitive notions such as the graph corresponding to the picture, and the west node of a cell, by fixing the meanings of the symbols used. Of course, language theoretically L_{graphs} is not any different from any other full language over an alphabet of the same cardinality.

If g is $\triangle$, $\triangleright$, $\triangledown$, $\triangleleft$, respectively, the node (x, g) is called the *north node*, *eat node*, *south node* or *west node*, respectively.

For every picture, we can find a corresponding graph. Also a weak converse is true: Let G be an arbitrary graph. Then there exists a picture p, and a set $S \subset (\mathrm{dom}(p) \times \Gamma)$ such that if we take S as the set of vertices and when $a, b \in S$, connect a to b if and only if there exists a path from a to b along zero or more vertices outside of S in the graph corresponding to p, then the graph obtained is isomorphic to G.

Theorem 2.63. *The language $L_{acyclic}$ is not in REC.*

Proof. Given n, we construct, for every total order $(\prec)$ of $[1, n]$, a picture $p_\prec$ of height n implementing a the total order $(\prec)$ on both sides: The picture p has odd width, and no vertical connections in the middle column. In addition, no edge travels backwards, that is, the node that is the endpoint of edges entering from the cell in direction d cannot have connections back to the cell in direction d. The west node of middle cell a of row A is connected to the east node of middle cell b of row B if and only if $A \prec B$. Symmetrically, the east node of a is connected to the west node of b if and only if $A \prec B$.

For every total order $(\prec)$, let us construct such a picture $p = p_\prec$. The middle column is over the single function f such that $f(\triangleright) = \{\triangleright\}$, $f(\triangleleft) = \{\triangleleft\}$ and $f(\triangle) = f(\triangledown) = \{\}$. Let x be the kth element of $[1, n]$ in the order $(\prec)$, and let a be the corresponding cell on the middle column. On the left side, we have a connection from the east side to the to the left, in the first k cells to the left of a. From the cell b reached, on column c, a path is drawn to the north and south from cell b, until it reaches the borders. From each node d on this path that lies on a row y such that $x \prec y$, a connection to the west side of column $c + 1$ is added. Furthermore, in every cell, on the left side of p, the east side of each cell is connected to its right neighbor. On the right side, the same picture is drawn with east and west flipped. The other nodes have no connections in the picture.

Consider a cycle C in this graph. Such a cycle cannot be contained on just one side of p: Consider a cycle on the left side of p. From any west node, all paths move to the middle column without turning. From any north or south node, all paths move down or up, respectively, until they reach the border, or turn east, reaching the west node of some cell. From any east node, a path will move to the left until it turns to the north or south. Therefore, in all cases the middle border is eventually reached.

Clearly the middle cell a is connected to the middle cell b on the left side of p if and only if $x \prec y$, where x is the row of a and y that of b, since by the previous paragraph, the only paths possible from the east node of a are ones that move left, turn up or down, and then return to the middle column. But the first turn must happen on the kth column to the left of a, and the second one must occur on a row r such that $x \prec r$ by the definition of p. The exact same holds for the right side.

Now, consider a cycle that uses nodes from both sides. Such a cycle must use a middle cell x_1. Without loss of generality, assume it starts on the east node of x_1. There are no edges up or down in the middle column, so the cycle must continue to the left side of p. This means it returns to the middle column in a cell x_2 such that the corresponding column is greater in the order $(\prec)$. Since the order is the same on both sides, this process continues. The cycle cannot close, since the middle cells x_i reached can never be smaller or equal to previous ones in the order $(\prec)$.

Now, let G be a hypothetical grammar for the language L of acyclic graphs. Then in particular, $p_\prec$ is in L for all $(\prec)$. We may assume all $p_\prec$ have the same width. There are $n!$ such pictures, all of height n, if $[1, n]$ is the set being ordered. Let us cut the pictures at (say) the left side of the middle column. Since for any m, $n! > (m^2)^n$ for large enough n, we may find pictures $p_\prec$ and $p_{\prec'}$ such that the sides can be swapped. But then, we obtain a picture $p' \in L$ where different orders $(\prec)$ and $(\prec')$ are used on both sides. In particular, $\exists a \neq b : a \prec b \wedge b \prec' a$. Thus, we obtain a (rectangular) cycle if we follow the connection from a to b on the $(\prec)$ side of p', and then the connection from b to a on the $(\prec')$ side of p'. □

Using vertical cuts is not always sufficient to prove a language outside REC. As an example of this, we give the following language.

Example 2.64. *The language L_{lt} of logarithmic twins consists of $2^n \times 2^n$ pictures over $\Sigma \cup \{\square\}$ containing two disjoint identical $n \times n$ squares over Σ, with $\square$ everywhere outside.*

The amount of information, measured in bits, that can be exchanged over a vertical cut line is $O(2^n)$. The information that needs to travel over a vertical cut consists of the contents of the two $n \times n$ squares, or parts of them, and information about their vertical position on the cut line, and the horizontal position of the vertical cut line on the squares.

At most $\Omega(n^2)$ information is needed to encode the contents of the square, the position on the vertical line is one of 2^n, and therefore requires $\Omega(n)$ bits, and the position of the cut on the square requires only $\Omega(\log n)$ bits. All in all, $\Omega(n^2)$ bits are needed. This implies that the lemma can not be used to prove L_{lt} non-recognizable using a vertical cut (or, symmetrically, a horizontal cut), since

$$\forall c : \exists d : \forall n : d2^n > cn^2.$$

It is, however, easy to see that L_{lt} is not in REC, by using a cut of a different shape.

Example 2.65. *The language L_{lt} is not in REC.*

Proof. Consider a $2^n \times 2^n$ square, and the $n \times n$ subsquares in its top left and bottom right corner. Let the cut E' be given by the set of edges out of the top left square. There are 2^{n^2} binary squares and the size of the cut is $4n$ with our canvas representation, since the subpicture is also cut out of the border. Consider the 2^{n^2} pictures obtained by having the two twins be at these corner squares. Since

$$2^{n^2} > ((m^2)^4)^n \text{ (see Appendix A).}$$

for any m and large enough n, one of the swaps of two such pictures would have to be in L_{lt}, but clearly this is not the case. □

There are still simple languages the lemma cannot — at least directly — prove non-recognizable, since we have to cut each of the pictures at the same set of edges. As a concrete example, consider the language similar to L_{lt}, but where each of the possible $n \times n$ squares is given its own position disjoint from the positions of others, and where the large square is large enough to allow this.

Consider an arbitrary cut C on the large square. If the smaller of the areas it cuts out contains k of the areas of the small squares, then the number of edges in C will be proportional to k. This means we run into the same problem as we did when trying to vertically cut L_{lt}: There is enough space for information to flow. For such cases, we need a stronger exchange lemma.

Definition 2.66. Let p^1 and p^2 be two labeled canvases with vertex-labelings l^1 and l^2 over Σ, with underlying canvases G^1 and G^2 both using the same set of directions. Let $C^i = (V_1^i, E'^i, V_2^i)$ be cuts of the G^i. For $i = 1, 2$ let S^i be the graph with vertex set

$$V_{S^i} = \{v \in V^i \mid \exists x : (v, x) \in E'^i \vee (x, v) \in E'^i\},$$

edge set $E_{S^i} = E'^i$, with the vertex labeling and road-coloring obtained as the restrictions of those of p^i. Let $h : S^1 \to S^2$ be a graph isomorphism that preserves sides and labels, and commutes with Γ-moves.

Let A and B be copies of $G^1 \cap V_1^1$ and $G^2 \cap V_2^2$, respectively, via injective graph homomorphisms $h_a : A \to G^1$ and $h_b : B \to G^2$ whose codomain restrictions to $G_i \cap V_i^i$ are graph isomorphisms. Then, we denote by $\mathrm{swap}(h, p^1, C^1, p^2, C^2)$ the labeled canvas $(V, E, r, \Sigma, \Gamma, f_V, f_E)$, where

$$V = A \cup B,$$

$$\forall v \in A : \forall g \in \Gamma : vg = \begin{cases} vg & \text{if } vg \text{ is defined in } A \\ h_b^{-1}(h(h_a(v))g) & \text{otherwise} \end{cases},$$

the outgoing edges from $v \in B$ are defined symmetrically and the vertex labeling function l is

$$l(v) = \begin{cases} l^1(h_a(v)) & \text{if } v \in A \\ l^2(h_b(v)) & \text{if } v \in B \end{cases}$$

We then have the following. The proof is essentially that of Lemma 2.59, and we omit it.

Lemma 2.67. *Let L be the language of a domino grammar with m allowed states, let $p^1, \ldots, p^n \in L$ be labeled canvases with the underlying unlabeled canvases $G^1, \ldots, G^n$, and let $C^i = (V_1^i, E'^i, V_2^i)$ be cuts of the G^i such that there exist isomorphisms between the S^i as in Definition 2.66. Then if all the swaps of the p_i with respect to the cuts C^i and isomorphisms chosen have their underlying shapes in $\mathcal{G}$, we have*

$$\exists j \neq j' : \mathit{swap}(h_{j,j'}, p^j, C^j, p^{j'}, C^{j'}) \in L \wedge \mathit{swap}(h_{j,j'}^{-1}, p^{j'}, C^{j'}, p^j, C^j) \in L$$

for some isomorphism $h_{j,j'}$.

Example 2.68. *For all n, we choose an order for the binary $n \times n$ squares. Consider the language L_{at} with pictures of size $(n * 2^{n^2} + 2) \times (n * 2^{n^2} + 2)$. Each picture of L_{at} must have two copies of a single $n \times n$ square t, on each of the sides of the picture (both one cell away from the edge of the picture), both with their top on row $2 + n * k$, if t is the kth $n \times n$ square in the ordering.*

The language L_{at} is not recognizable.

Proof. For some n, take all the $n \times n$ possible pictures of size $(n * 2^{n^2} + 2) \times (n * 2^{n^2} + 2)$, and consider the cuts that remove the leftmost nonempty $n \times n$ square from the large square. Again, for two different pictures and cuts, we get a repetition of border states, and the small squares can be exchanged, producing a picture not in L_{at}. □

In an example in the first first chapter, we saw that the recognizable subsets of L_{words} are just the picture representations of the regular languages. The following characterizes the unbordered case. We only give a proof scetch.

Theorem 2.69. *The intersections of $\mathbb{Z}^2$-REC languages with the unbordered representations of languages in L_{words}, over any alphabet Σ, are exactly languages over Σ whose complement is recursively enumerable.*

Proof. It is clear that the complement must be recursively enumerable for any such language, since if there does not exist a tiling of the plane around the word, there must exist a finite ball around it that cannot be tiled, which a Turing machine enumerating the tilings will eventually find.

For the other direction, given a language L with a recursively enumerable complement, let A be a Turing machine that accepts exactly the words of L^c, and never halts on other words. Then, we run this Turing machine in the half-plane under the given word using a standard tiling representation. A tile representing an accepting state can not have another tile under it, making completing the tiling impossible, and otherwise a tiling exists for the whole plane with blank tiles above the input, and the infinite run on a bi-infinite tape under it. □

2.4 Characterization using logic formulas

In this chapter, we characterize REC using logic formulas. This is a much less obvious result than the equivalence of projections of LOC and languages of domino grammars. The results of this section are essentially from [6], although the approach is rather different. The chapter is self-contained, and unlike [6], does not refer to external sources. The notable difference is that we prove the result FO = LTT using closure properties of LTT instead of inductively proving a characterization of sets of models of first order formulas, and obtain a slightly more general result.

First, let us define LTT, an intermediate class between LOC and REC that has a more natural logical characterization (and is perhaps more natural than LOC as a picture class as well).

The intermediate grammar is based on the concept of threshold counting. When threshold counting, one keeps exact count up to a threshold, but only remembers that the threshold has been exceeded for larger numbers.

Definition 2.70. Let $T \in \mathcal{T}, k > 0$. Then we write $p \overset{k,T}{\sim} p'$ if $\forall t \in T$, either

$$c(p,t) = c(p',t)$$

or

$$c(p,t) \geq k \wedge c(p',t) \geq k.$$

Definition 2.71. A locally threshold testable grammar G is a tuple (Σ, T, k, X), where $T \subset \mathcal{T}, |T| < \infty, k > 0$, and X is a set of $\overset{k,T}{\sim}$-equivalence classes. The *locally threshold testable language* $\mathcal{L}(G)$ associated with G is the set of pictures $p \in \Sigma_{**}^{*}$ in one of the equivalence classes X. We denote the class of locally threshold testable languages by LTT.

We may assume an LTT grammar only counts occurrences of tiles of a single rectangular shape, by extending each non-rectangular tile into a rectangle in all possible ways.

The following is also a rather simple corollary of DFA $\subset$ REC, proved in Section 3.3. It is also a direct corollary of LTT $\subset$ DREC, which is proven in Section 2.5.

Theorem 2.72. *LTT $\subset$ REC.*

Proof. Let the tiles T define the LTT grammar G, and let B be the ball of size $\max\{d((0,0),x) \mid t \in T, x \in \mathrm{dom}(t)\}$ around the origin. We first use Lemma 2.39 to compute the relation R_1 adding the information in the neighborhood B around each position to the states of p.

Next, given a set $S \subset [0,k]^T$, we compute another incremental relation $R_2 \subset \Sigma_{**}^{*} \overset{\times}{\to} ([0,k]^T \times [0,k]^T)_{**}^{*}$. Here, $\overset{\times}{\to}$ means $\times$, but implies δ is thought of as a multivalued function. The right sides computed on the new layer only depend on the size of p. The relation is defined such that if $(p, p') \in R_2$, then

$$p'[(1,1)]_1 = (\forall x : x \mapsto 0)$$

$$(i,j) \in \mathrm{dom}(p'), j < |p'| \implies p'[(i,j+1)]_1 = p'[(i,j)]_2$$

$$(i,j) \in \mathrm{dom}(p'), j = |p'|, i < \underline{\overline{p'}} \implies p'[(i+1,1)]_1 = p'[(i,j)]_2$$

$$p'[(\underline{\overline{p'}}, |p'|)]_2 \in S$$

Now, using all the added information, we can locally increment the counters correctly, since the counter layers of the cells have been linked into a chain by R_2. We then check at each position $x \in \mathrm{dom}(p)$ that

$$g(t) = \begin{cases} f(t)+1 & \text{if } p[x + \mathrm{dom}(t)] \sim t \\ f(t) & \text{otherwise} \end{cases},$$

where (f, g) is the state added by R_2 on top of x, and where addition is done with threshold k. Note that $p[x + \mathrm{dom}(t)] \sim t$ can be checked due to the information added by R_1.

This gives us the required REC grammar, when S is chosen as the set of allowed combinations of amounts of tiles in G. □

Corollary 2.73. *REC is the class of languages obtained by taking the symbol-to-symbol projections of languages in LTT.*

Proof. Since REC is closed under such projections, the symbol-to-symbol projections of languages in LTT are a subclass of REC. Conversely, since LTT is a superclass of LOC, and REC is the class of symbol-to-symbol projections of LOC, in particular projections of LTT contain REC. □

We use the definitions of [1] for the classes FO and EMSO. In order to state the definitions, we need some basic definitions from first and second order logic.

Definition 2.74. A *relational structure* of type $(a_1, \ldots, a_n)$, where $\forall i : a_i \in \mathbb{N}$ is an $(n+1)$-tuple $(X, R_1, \ldots, R_n)$ such that $\forall i : R_i \subset X^{a_i}$. The relational structures of type $(a_1, \ldots, a_n)$ are denoted by $\mathcal{R}(a_1, \ldots, a_n)$.

Definition 2.75. For $p \in \Sigma_*^*$, the relational structure $\mathcal{R}(p)$ corresponding to p is the structure $(\mathrm{dom}(p), S_\sim, S_\triangleright, S_\triangledown, (P_a)_{a\in\Sigma})$ of type $(2, 2, 2, \underbrace{1, \ldots, 1}_{|\Sigma|})$ defined as

$$(x, y) \in S_\sim \iff x = y$$

$$(x, y) \in S_\triangleright \iff x + (0, 1) = y$$

$$(x, y) \in S_\triangledown \iff x + (1, 0) = y$$

$$x \in P_a \iff p[x] = a$$

We take a set $\mathcal{X}_{fo} = x_1, x_2, \ldots$ of *first-order variables*, which take their values in $\mathrm{dom}(p)$. We also consider *monadic second-order variables* $\mathcal{X}_{mso} = X_1, X_2, \ldots$, which take their values in $2^{\mathrm{dom}(p)}$. We build formulas out of variables, quantifiers, boolean connectives and relations, as defined in the following sequence of definitions.

Definition 2.76. The *atomic formulas* of type $(a_1, \ldots, a_n)$ are exactly

$$\{R_i(x_1, \ldots, x_{a_i}) \mid x_j \in \mathcal{X}_{fo}, 1 \le i \le n\} \cup \{X(x) \mid x \in \mathcal{X}_{fo}, X \in \mathcal{X}_{mso}\}.$$

In the case of formulas defining picture languages, we write $x \sim y$, $x \triangleright y$, $x \triangledown y$ and $P_a(x)$ instead, to make formulas more readable.

Definition 2.77. The *formulas* of type $(a_1, \ldots, a_n)$ are given by the following

- All atomic formulas are formulas.
- If ϕ_1 and ϕ_2 are formulas, and they do not quantify over a common variable x, then $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$ and $\neg\phi_1$ are formulas.
- If $\phi(x)$ is a formula, with $x \in \mathcal{X}_{fo}$, then $(\exists x)\phi$ and $(\forall x)\phi$ are formulas.
- If $\phi(X)$ is a formula, with $X \in \mathcal{X}_{mso}$, then $(\exists X)\phi$ and $(\forall X)\phi$ are formulas.

The free variables of a formula are the variables the occur outside the scope of a quantifier. By $\phi(x)$, we denote a formula where $x \in \mathcal{X}_{fo}$ occurs as a free variable and is not quantified anywhere in ϕ. Similarly, $\phi(X)$ is a formula where $X \in \mathcal{X}_{mso}$ occurs as a free variable and is not quantified anywhere in ϕ.

Note that no variable can be quantified twice within the same formula.

We may also think of a formula as having free variables it does not use. This is used in the following definition, where the sets of free variables of two formulas are extended to a common set of free variables.

Definition 2.78. Given a formula $\phi(X_1, \ldots, X_k, x_1, \ldots, x_m)$ with free variables $(X_i), (x_i)$ of type $(a_1, \ldots, a_n)$, and a relational structure R of the same type, we say $(R, (B_i)_i, (b_i)_i)$ is a quasimodel of ϕ, if the formula is satisfied for the relational structure, and the sets of values B_i and b_i for the free variables.

That is, define

$$P = P(k, m) = \{((X, (R_i)_i), B_1, \ldots, B_k, b_1, \ldots, b_m) \in \mathcal{R}(a_1, \ldots, a_n) \times (2^X)^k \times X^m : \forall i : B_i \subset X, b_i \in X\},$$

We then define the set of quasimodels of ϕ to be the subset $\mathcal{L}(\phi) \subset P$ defined inductively by

$$\mathcal{L}(X_i(x_j)) = \{((X, (R_i)), B, b) \in P(1, 1) \mid b \in B\}$$

$$\mathcal{L}(R_j(X_1, \ldots, X_k, x_1, \ldots, x_m)) = \{((X, (R_i)), (B_i), (b_i)) \in P(k, m) \mid R_j(B_1, \ldots, B_k, b_1, \ldots, b_m)\}$$

Let $\phi = \phi_1 \wedge \phi_2$, where ϕ, ϕ_1 and ϕ_2 have the same free variables $X_1, \ldots, X_k, x_1, \ldots, x_m$, possibly by extending the sets of variables in ϕ_1 and ϕ_2. Then

$$\mathcal{L}(\phi_1 \wedge \phi_2) = \mathcal{L}(\phi_1) \cap \mathcal{L}(\phi_2).$$

Similarly to the case $\phi = \phi_1 \wedge \phi_2$,

$$\mathcal{L}(\phi_1 \vee \phi_2) = \mathcal{L}(\phi_1) \cup \mathcal{L}(\phi_2).$$

Let $\phi = \neg\phi_1$ with free variables $X_1, \ldots, X_k, x_1, \ldots, x_m$. Then for $R \in P(k, m)$,

$$R \in \mathcal{L}(\phi) \iff R \notin \mathcal{L}(\phi_1).$$

Let $\phi = (\exists x_j)\phi_1$, and let ϕ_1 have free variables $X_1, \ldots, X_k, x_1, \ldots, x_m$. Then

$$\mathcal{L}(\phi) = \{((X, (R_i)), (B_i), b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_m) \in P(k, m-1) \mid \exists b_j \in X : ((X, (R_i)), (B_i), (b_i)) \in \mathcal{L}(\phi_1)\}$$

Let $\phi = (\forall x_j)\phi_1$, and let ϕ_1 have free variables $X_1, \ldots, X_k, x_1, \ldots, x_m$. Then

$$\mathcal{L}(\phi) = \{((X, (R_i)), (B_i), b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_m) \in P(k, m-1) \mid \forall b_j \in X : ((X, (R_i)), (B_i), (b_i)) \in \mathcal{L}(\phi_1)\}$$

and second order quantification is defined similarly.

We call the quasimodels of a formula without free variables *models*.

Now, we can define the logic based picture classes considered in this chapter. We say a formula is a *first-order formula* if no quantification is done over second order variables. We say a formula is an *existential monadic second order formula* if all quantification over second order variables is done with an existential quantifier, and all the quantification happens at the top level of the formula. That is, the existential monadic second order formulas are of the form $\exists X_1, \ldots, X_k : \psi(X_1, \ldots, X_k)$, where ψ is a first order formula.

Definition 2.79. Given a formula ϕ without free variables, we denote by $\mathcal{L}(\phi)$ the set of pictures p such that $\mathcal{R}(p)$ is a model of ϕ. Such languages, where ϕ is a first-order formula, give the picture class FO. The corresponding class for existential monadic second order formulas is called EMSO.

Our main goal in this section is to prove the following theorem from [6].

Theorem 2.80. *EMSO = REC.*

As in [6], we will first to prove the following result, which is interesting in itself. The theorem EMSO = REC then easily follows.

Theorem 2.81. *FO = LTT.*

We do not prove FO = LTT directly. Instead, we first show LTT $\subset$ FO, and then define an auxiliary picture class FOC for which we show FO $\subset$ FOC $\subset$ LTT. The class FOC is defined by its closure properties, which can easily be used to implement first-order formulas. It is then enough to show LTT has these closure properties as well. The first proof is simple.

Lemma 2.82. *LTT $\subset$ FO*

Proof. It is enough to, given $k \geq 1$, and a *rectangular* tile $T \in \mathcal{T}_{\Sigma\cup\{\#\}}$, construct a first order formula $\phi(k,T)$ that is true if and only if T occurs in p at least k times. Any LTT language is a boolean combination of such formulas: To test if a pattern occurs exactly j times, we use the formula $\phi!(k,T) = \phi(k,T) \wedge \neg\phi(k+1,T)$. Then, every LTT grammar G is essentially a boolean combination of $\phi(k,T)$ and $\phi!(k,T)$, for different k and T, since we may assume G only contains rectangular patterns.

Let T be of shape $m \times n$. First assume it contains no #. The formula $\phi(k,T)$ uses the $k|\mathrm{dom}(T)|$ variables

$$\{x^s_{(i,j))} \mid 1 \leq s \leq k, (i,j) \in [1,m] \times [1,n]\}$$

Our goal is to write a formula that, given $p \in \Sigma^*_*$, posits the existence of positions for the variables $x^s_{(i,j)}$ such that, for each s,

$$\exists x \in \mathrm{dom}(p) : \forall (i,j) \in [1,m] \times [1,n] : x^s_{(i,j)} = x + (i,j),$$

that

$$\forall (1 \leq s < s' \leq k) : x^s_{(1,1)} \neq x^{s'}_{(1,1)},$$

and that

$$\forall s,i,j : p[x^s_{(i,j)}] = T((i,j))$$

This simply means we guess the positions of k different tiles similar to T on the domain of the picture. This is clearly possible if and only if there are at least k subtiles similar to T in p. Note that here we need the fact T may not contain symbols of #, since variables cannot quantify over positions outside the picture.

All of the required properties are readily translatable into a first order formula, and we give one explicitly. To guess the contents of variable, we use $(\exists_{i,j,s} x^s_{(i,j)}) = (\exists x^1_{(1,1)})(\exists x^1_{(1,2)})...(\exists x^k_{(m,n)})$. We construct three formulas ψ, ξ and κ, checking the three constraints on the assignment of variables.

We call the formula checking the shape of tiles ψ, and it is given by

$$\psi = \bigwedge_{1 \leq s \leq k} \psi_{\triangleright}(s) \wedge \psi_{\triangledown}(s),$$

where

$$\psi_{\triangleright}(s) = \bigwedge_{(i,j) \in [1,m] \times [1,n-1]} \psi_{\triangleright}(s,i,j)$$

$$\psi_{\triangledown}(s) = \bigwedge_{(i,j) \in [1,m-1] \times [1,n]} \psi_{\triangledown}(s,i,j)$$

$$\psi_{\triangleright}(s,i,j) = x^s_{(i,j)} \triangleright x^s_{(i,j+1)}$$

$$\psi_{\triangledown}(s,i,j) = x^s_{(i,j)} \triangledown x^s_{(i+1,j)}.$$

The formula checking all tiles have their top left corners in different positions is simply

$$\xi = \bigwedge_{1 \leq s < s' \leq k} \neg(x^s_{(1,1)} \sim x^{s'}_{(1,1)}).$$

Finally, to check that all tiles have the correct contents, we use

$$\kappa = \bigwedge_{s,i,j} P_{T((i,j))}(x^s_{(i,j)}).$$

Then, the complete formula is

$$\phi(k,T) = (\exists_{i,j,s} x^s_{(i,j)})(\psi \wedge \xi \wedge \kappa).$$

As for tiles T containing #, we note that T must be a tile T' over Σ surrounded by #. Similarly to the case of no #, we use variables to mark the domains of the T' occurring in the picture. the only difference is we need to check that borders occur on the correct sides. But this is easy: for instance, to make sure the south neighbor of x is #, we can use the formula $\neg(\exists y)(x \triangledown y)$. □

Now, let us proceed by introducing the class FOC. It is defined by using closure properties, so let us first introduce the operations it is closed under.

Definition 2.83. The *variable languages* are

$$L_{single}(x) = \{p \in \{0,x\}^*_* \mid c(p,x) = 1\},$$

where x is an arbitrary symbol. We write $x(i,j)$ to refer to the language of pictures $p \in L_{single}(x)$ such that $p[i,j] = x$.

Definition 2.84. Given $L \subset \Sigma^*_*$, we write $L(x)$ for $L \times L_{single}(x)$. The function $L \mapsto L(x)$ is called the *variable introduction* operation.

For a short explanation of the subtleties behind 'Let $L \subset \Sigma^*_*$ such that $\Sigma = \Sigma' \times \{0,x\}$', see the end of Section 2.2.

Definition 2.85. Let $L \subset \Sigma^*_*$ such that $\Sigma = \Sigma' \times \{0,x\}$. Then, we say x *is a variable*, if

$$\forall p \in L : \pi_2(p) \in L_{single}(x).$$

Definition 2.86. Let $L \subset \Sigma_*^*$ such that $\Sigma = \Sigma' \times \{0, x\}$, where x is a variable. Then, we write

$$(\triangledown x)(L) = \pi_1(L),$$

called the *existential quantification* operation. We write

$$(\triangle x)(L) = \{\pi_1(p) \mid p \in L, \forall (i,j) \in \mathrm{dom}(p) : p \times x(i,j) \in L\}$$

for the *universal quantification* operation.

Definition 2.87. If $p \in (\Sigma \times \{0, x\})_*^*$, and x is a variable, we write

$$p[x] = v \in \mathbb{Z}^2 \text{ such that } p[v] = x.$$

Note that this is a kind of reverse indexing (or search) operation even though it uses the indexing notation.

Definition 2.88. Let $L \subset (\Sigma \times \{0, x\})_*^*$, where x is a variable. Then the *symbol check* operations are

$$L(x = a) = \{p \in L \mid p[p[x]] = (a, x)\},$$

where $a \in \Sigma$

Definition 2.89. Let $L \subset (\Sigma \times \{0, x\} \times \{0, y\})_*^*$, where both x and y are variables. Then the *adjacency* operators are

$$L(x \sim y) = \{p \in L \mid p[x] = p[y]\},$$

$$L(x \triangleright y) = \{p \in L \mid p[x] + (0, 1) = p[y]\}$$

and

$$L(x \triangledown y) = \{p \in L \mid p[x] + (1, 0) = p[y]\}.$$

The variable introduction operation adds a variable 'layer' on top of the pictures, and the symbol check and adjacency operators check a local property of pictures with variables on top of them, but do not erase the variables. The quantification operators remove a variable layer, and the universal quantification operator additionally removes the pictures where the variable did not occur in every position. We use these operations, and the boolean operations, to define FOC.

Definition 2.90. FOC is the smallest class that contains the full languages, and is closed under the variable introduction operation, the existential and universal quantification operations, the symbol check operation, the adjacency operations, and the operations $\cap$, $\cup$ and $\neg$.

The definitions of the operations do not really give us a notation for writing longer expressions, since how the operation is applied depends on the decomposition of Σ used. However, if Σ is a cartesian product $P_1 \times \cdots \times P_n$, where $P_k = \{0, x\}$ is one of the terms, and $x \notin P_j$ for $j \neq k$, then we take $(\triangledown x)(L)$ to mean existential quantification with respect to the decomposition $\Sigma = (\prod_{j \neq k} P_j) \times P_k$. Then, it is easy to write expressions defining languages in FOC, as long as we never do any 'renaming'.

We do not give a formal proof for FO $\subset$ FOC, but just give an example of converting an FO formula into an equivalent FOC expression. It should be evident that the same idea works in general: We mark the places of first order variables using the language theoretic variables, as defined above. There will then be a one-to-one correspondence between the quasimodels of the subformulas and the FOC languages corresponding to them, which could of course be formally proved by a lengthy induction argument.

Example 2.91. *Consider the FO language $L \subset \Sigma_*^*$, where $\Sigma = \{a, b, c, d\}$, defined by the formula*

$$\phi = (\exists x)(\exists y)(x \triangleright y \wedge P_a(x) \wedge P_b(y)) \vee (\forall z)(P_c(z) \vee P_d(z)).$$

This is the language of pictures that either contain $\begin{array}{|c|c|}\hline a & b \\ \hline \end{array}$ *or are completely over $\{c, d\}$.*

We define the two FOC languages $U_{x,y} = \Sigma_*^*(x)(y)$ and $U_z = \Sigma_*^*(z)$. Then the expression

$$(\triangledown x)(\triangledown y)(U_{x,y}(x \triangleright y) \cap U_{x,y}(x = a) \cap U_{x,y}(y = b)) \cup (\triangle z)(U_z(z = c) \cup U_z(z = d))$$

shows the language L is also in FOC: The language $U_{x,y}(x \triangleright y) \cap U_{x,y}(x = a) \cap U_{x,y}(y = b)$ contains exactly the pictures where x and y mark an occurrence of $\begin{array}{|c|c|}\hline a & b \\ \hline \end{array}$*, so applying $(\triangledown x)(\triangledown y)$ gives us exactly the pictures over Σ containing the pattern.*

The language $U_z(z = c) \cup U_z(z = d)$ on the other hand contains all pictures where variable z is on top of either a c or a d. Therefore, applying $(\triangle z)$ gives us the language of pictures p such that for any position $(i, j) \in dom(p)$, $p \times z(i, j)$ occurs in $U_z(z = c) \cup U_z(z = d)$, that is, all positions (i, j) of p contain either a c or a d.

Let us now show LTT has all the closure properties of FOC as well, implying FOC $\subset$ LTT. Note that, in order to prove a picture language is in LTT, all we need to prove is that there exists a finite set of tiles such that knowing the number of occurrences of each of them is enough to determine whether the picture is in the language. The rule that determines whether a picture belongs to the language, given the threshold counted amounts of the tiles the grammar is interested in, is again called the *local rule*. The local rule is of course simply a look-up table, but it is often nicer to give an algorithm that computes the table, just like in the case of LOC.

Theorem 2.92. *LTT is closed under the boolean operations.*

Proof. In the union or intersection L of $L_1 = \mathcal{L}(G_1)$ and $L_2 = \mathcal{L}(G_2)$, we may assume G_1 and G_2 use the same sets of tiles and the same threshold. We construct an LTT grammar G for L. The local rule of G can determine whether the picture belongs to L_1, and whether it belongs to L_2 by computing the amounts of the same tiles as the two grammars, and therefore it is clear that it can also check if the picture belongs to L. As for negation, the local rule of grammar G may compute the output of the original local rule, and flip the output to obtain $\mathcal{L}(G)^c$. □

So note that even though REC is not closed under complementation, its subclass LTT is closed under this operation.

Theorem 2.93. *LTT is closed under the variable introduction operation.*

Proof. Let $L = \mathcal{L}(G)$ be an arbitrary LTT language over Σ, and let x be the variable added. The local rule for G' such that $\mathcal{L}(G') = L(x)$ counts all tiles T' of the same shapes as the tiles T of G, and the amounts of 1×1 tiles. To obtain the amount of occurrences of tile $t \in T$ in p, it sums the amounts of occurrences of tiles $t' \in T'$ such that $\pi_1(t') = t$. The 1×1 tiles can be used to check x occurs exactly once on the second layer by summing up the amounts of occurrences of tiles t' with $\pi_2(t') = x$ and checking this sum is 1. □

Theorem 2.94. *LTT is closed under the symbol check operation.*

Proof. Let $L = \mathcal{L}(G)$ be an LTT language over $\Sigma \times \{0, x\}$ such that x is a variable in L. The grammar G' for the symbol check of L with respect to x and symbol $a \in \Sigma$, given p, first checks that p belongs to L as in the proof of the previous theorem. It then computes the amounts of occurrences of different 1×1 tiles, and checks that all such tiles either have a 0 on the variable layer, or have an a on the Σ layer. □

Theorem 2.95. *LTT is closed under the adjacency operation.*

Proof. Similarly to the symbol check operation, it is enough to check that the picture is in the original language, then count the amounts of different 1×2 (or 2×1) tiles to check the variables occur next to each other in the correct order. □

Theorem 2.96. *LTT is closed under the existential and universal quantification operations.*

Proof. Let $L = \mathcal{L}(G)$ be an LTT language over $\Sigma \times \{0, x\}$ such that x is a variable in L. Let m be such that G determines whether $p \in (\Sigma \times \{0, x\})^*_*$ belongs to L based on the amounts of different $m \times m$ tiles, and let k be the threshold of G. Let $L' = \triangledown L$. We need to construct a grammar G' that determines whether, given picture $p \in \Sigma^*_*$, there exists a position $(i, j) \in \mathrm{dom}(p)$ such that $p \times x((i, j)) \in L$.

The grammar G' uses the $(2m-1) \times (2m-1)$ tiles such that the middle cell contains a symbol of Σ, and threshold $k' = k + m^2$. If x is added on top of one of the cells of p, this cell is the center cell of some $(2m-1) \times (2m-1)$ tile. Now let the size $(2m-1) \times (2m-1)$ tile t occur n times in picture $p \in \Sigma^*_*$. Then if x were added on top of this tile, the $m \times m$ tiles affected would be exactly the size $m \times m$ subtiles of the occurrance of t on p where the middle cell (i, j) is replaced with x.

Thus, we are able obtain the amounts of different $m \times m$ subtiles of $p \times x(i, j)$: The tiles containing an x are the m^2 subtiles t' of t with x in the correct cell, and the amounts of the rest of the tiles are obtained by computing the amounts of m^2 tiles occurring in p, and subtracting the ones x was added on. Note that an x can occur on one type of tile at most m^2 times, so the threshold of $k + m^2$ is enough to obtain the threshold count for the pictures of $p \times x(i, j)$.

The case of $\triangledown L$ is similar, now the local rule simply checks the x could've been added as the middle cell of any of the $(2m-1) \times (2m-1)$ tiles occurring in p. □

Since also $\Sigma_*^* \in$ LTT for any alphabet Σ (by an empty grammar). The theorems proved above imply the following as a corollary, also concluding the proof of FO = LTT.

Corollary 2.97. *FO* $\subset$ *FOC* $\subset$ *LTT.*

We can now prove the main theorem of this chapter, namely that the classes EMSO and REC are equal. We split the proof in two cases.

Lemma 2.98. *EMSO* $\subset$ *REC.*

Proof. Let $\psi = (\exists X_1)\cdots(\exists X_k)\phi$ be an arbitrary EMSO formula, and let L be its language over Σ. First given a picture $p \in \Sigma_*^*$, let a relation R_1 add, on each cell of p, and arbitrary element of 2^k, obtaining a picture p'. Substitute $\bigvee_{v\in 2^k} P_{(a,v)}(x)$ for each occurrence of $P_a(x)$ in ϕ, and substitute $\bigvee_{a\in\Sigma, v\in 2^k, v_j=1} P_{(a,v)}(x)$ for each occurrence of $X_j(x)$ to obtain a first-order formula ϕ'. Consider the set of models $\mathcal{L}(\phi')$ for this new formula. Clearly, the possible vectors of $\{0,1\}^k$ on the cells correspond to possible contents of the variables X_j.

Since there exists an LTT grammar for $\mathcal{L}(\phi')$, there also exists an REC grammar G for it. Now consider a picture $p \in L$. There exist values for the X_i such that the first-order formula holds true. We may then choose the elements 2^k corresponding to the values of X_i as the ones added by R_1. The picture p' obtained this way is in $\mathcal{L}(G)$, since $p' \in \mathcal{L}(\phi')$. Conversely, if for some assignment of elements of 2^k on top of picture p, the picture p' obtained is accepted by G, then we may take the sets X_i corresponding to the elements of 2^k added as those chosen by the existential quantifiers. It is then clear that the relation R_1 and the grammar G can be used to obtain the required REC grammar. □

Lemma 2.99. *REC* $\subset$ *EMSO.*

Proof. Let the states of the REC grammar G be Q, and assume without loss of generality that $|Q| = 2^k$. We take $X_1, \ldots, X_k$ as the existentially quantified second-order variables. For each position $(i,j) \in X$, in the relational structure $\mathcal{R}(p)$, vectors $(X_1((i,j)), \ldots, X_k((i,j)))$ will correspond to a state of Q on top of position (i,j) of p, using some bijection f between vectors of 2^k and Q.

Since LTT $\subset$ FO, we may check that the assignment of values for the variables X_i corresponds to a valid set of states for each position of the picture p. In fact, to check the horizontal tile $\boxed{a \mid b}$ not containing a # does not occur in the picture, we may use

$$(\forall x)(\forall y)((x \triangleright y) \rightarrow ((\bigvee_{z\in\Sigma-\{a\}} x = z) \vee (\bigvee_{z\in\Sigma-\{b\}} y = z)),$$

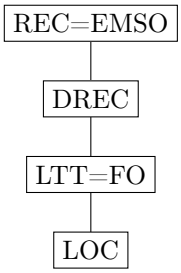
and to check a never occurs at the right border, we may use

$$(\forall x)(P_a(x) \rightarrow (\exists y)(x \triangleright y),$$

where in both formulas, $\phi_1 \rightarrow \phi_2 = \neg\phi_1 \vee \phi_2$. □

This concludes the proof of EMSO = REC

Figure 4: A Hasse diagram depicting REC and its relatives introduced in this chapter. All the inclusions are proper.



2.5 DREC

In this section, we define a deterministic variant of REC, and build a very simple inclusion lattice (which happens to be a chain). This is depicted in Figure 4.

Definition 2.100. The DREC grammars and their languages are just domino grammars (Σ, Δ, R, T) with the additional property of north-west determinism. This means that if the current symbol and the states of the north and west neighbors are known, then only one state is allowed for the current cell by both R and T. We say a state s is *final* if it can occur as the state of the bottom right cell, that is, $\{\begin{array}{|c|c|}\hline s & \# \\ \hline \end{array}, \begin{array}{|c|}\hline s \\ \hline \# \\ \hline \end{array}\} \subset T$.

It should be clear that just like REC, also DREC is closed under unions, intersections and cartesian products. In all cases we can take as the set of states of the new grammar the cartesian product of the state sets of the two languages. In the boolean functions, the states are assigned based on the symbols of the single picture, and in the cartesian product case, the two layers of the symbols are given to their respective grammars. In union, it is checked that one grammar accepts. In intersection and cartesian product, both must accept. The determinism of the new grammar follows directly from the determinism of the original grammars.

Note that for union in the general REC case, a very simple solution is obtained by taking disjoint state sets for the two grammars, and using their union as the set of states, with the obvious local rule. However, in the deterministic case, the top left symbol of the picture would then need to determine which language the picture belongs to, which is not possible in general.

Recognizing a picture with a DREC grammar can be thought of a deterministic process assigning states in a diagonal wave over the picture. This intuition is useful also in finding languages outside DREC: Information flow is rather constrained, so it is visually clear that a DREC grammar will not have time to compare, say, the rightmost column and the bottom row for equality. We will shortly prove this language to be outside DREC.

Often, the closure of DREC under rotations is taken as the definition. Let us temporarily call this class DRECR. It should be clear that DRECR is the picture class of languages accepted by tile sets that are either north-west, north-east, south-west or south-east deterministic.

Superficially, taking the closure makes it harder to find a language outside the class. However, just taking a closure after the fact (without the class somehow having a use for the new languages obtained) does not really strengthen a class. Let us give a more concrete justification for not taking the closure. The crucial thing to note is that DRECR is not closed under cartesian products as a rather direct consequence of the fact DREC is not closed under rotation.

In the following, we take it as a given that DREC $\subsetneq$ DRECR. The existence of the L in the assumption of the theorem will be proven shortly.

Lemma 2.101. *If L is a language in DRECR – DREC, then DRECR is not closed under cartesian products.*

Proof. Suppose on the contrary that DRECR is closed under cartesian products. Let L be over Σ such that $1 \notin \Sigma$. We take the union $L' = L \cup \{1\}^*_*$, and note that also L' must be in DRECR, since the tileset can choose whether to use the grammar of $\{1\}^*_*$ or the grammar of L after seeing the first symbol. Now let L'' be the cartesian product of the four rotations of L'. There must exist a DRECR grammar for it by the assumption, and thus a tile set that is deterministic in one of the diagonals. By symmetry, we may assume L'' is in DREC. But then also the intersection with $(\Sigma \times \{1\}^3)^*_*$ is in DREC, since DREC is closed under intersection. Then, since DREC is a picture class, we may rename $(a, 1, 1, 1)$ to a for all $a \in \Sigma$, and obtain that L is in DREC, which is a contradiction. □

This implies the following. The usefulness of the result comes from the fact almost every sensible class is closed under rotations and cartesian products.

Theorem 2.102. *Let L be a language in DRECR – DREC and let CLS be a picture class closed under rotations and cartesian products. Then DREC $\subset$ CLS $\implies$ DRECR $\subsetneq$ CLS.*

Proof. DREC $\subset$ CLS implies DRECR $\subset$ CLS because CLS is closed under rotation, and since $L \in$ DRECR – DREC, the proper inclusion follows from the fact DRECR is not closed under cartesian products. □

Let us begin the building of the lattice.

Theorem 2.103. *LOC $\subsetneq$ LTT.*

Proof. Clearly LOC is a subset of LTT, for given a LOC grammar, we can simply count the occurrences of all forbidden patterns, and accept if none occur even once.

Consider the language $L = \{a, b\}^*_* \cup \{a, c\}^*_* \cup \{b, c\}^*_*$. This language is in LTT by a grammar that checks that of the three letters, at most 2 occur in the picture. However, there cannot be a LOC grammar for it. On the contrary, suppose G were such a grammar. Let the diameter of tiles in G be k. Then, the picture $a^k b^k c^k$ is not in L, and thus not in $\mathcal{L}(G)$ either, so there must be a forbidden tile t that occurs in it. But t cannot contain all three letters. Therefore either the word $a^k b^k$ or the word $b^k c^k$ contains t as well. □

Theorem 2.104. *LTT $\subset$ DREC.*

Proof. Let G be an LTT grammar, and let the grammar G count only rectangles of size k.

We construct a domino grammar that carries, in the state of each cell, the contents of the square of width and height k whose bottom right corner is the cell in question. This includes the # cells in the square. This is clearly possible to do north-west deterministically, since we can access the contents of the two squares of the same shape rooted in the north and west neighbors, and we can access the current symbol. In the following, we think of the # east and south of the picture as having the symbol $\#'$ to simplify the discussion.

On each row, a counter is carried from left to right. It counts, up to the threshold, the numbers of occurrences of the different square tiles of size $k \times k$ seen sofar. On the last row of the picture, we want slightly more information, namely the amounts of tiles with $\#'$ in some j of the bottom rows and $\Sigma \cup \{\#\}$ on the other rows. The last row cannot be detected, so this computation is done on every row. Since the grammar knows the amounts of rectangles seen sofar, of all sizes $i \times k$ for $1 \leq i < k$, this information can easily be obtained.

On each column, a second counter moves downward, combining all the information about $k \times k$ tiles into a single counter. This counter will always hold, at a given cell, the numbers of $k \times k$ patterns that occur in the whole subpicture northwest from the current cell, since the threshold counting monoids $\langle a \mid a^k = a^{k+1} \rangle$ are commutative. On the last column, we also need to compute the amounts of tiles with $\#'$ on some j of the rightmost columns. Again, we must compute this information on every row.

Now, let us choose the final states. In the corner cell we know the amount, up to a threshold, of every pattern containing no $\#'$, containing $\#'$ only on some of the last rows, or containing $\#'$ only on some of the last columns. The amounts of tiles with a $\#'$ in both the top right cell and the bottom left cell are then obtained based on the current $k \times k$ square rooted in this cell, and thus we may check if G would have accepted this combination of amounts of different $k \times k$ tiles, which concludes the construction. □

In order to show the DREC is a proper subset of REC, we need to prove the existence of L in Lemma 2.101. We also obtain the proper inclusion of LOC in DREC as a corollary.

Lemma 2.105. *DREC is not closed under clockwise rotations. In particular,* $L_{t=r} = \{p \in \Sigma_*^* \mid p[1,*] = p[*,|p|]^R\}$ *is in DREC, while its clockwise rotation is not.*

Proof. Consider the language $L_{t=r}$ and its clockwise rotation $L_{b=r} = \{p \in \Sigma_*^* \mid p[\underline{p},*] = p[*,|p|]\}$. We will prove that $L_{t=r}$ is in DREC, while $L_{b=r}$ is not.

First, let us construct a DREC grammar for $L_{t=r}$. The grammar sends signals diagonally, comparing each symbol of the top row to the corresponding column symbol south-east of it. States of the top row read the contents of the current row, and other cells read this information from their north-west neighbor (thus propagating the read symbol south-east), if this neighbor is carrying such information. No such information is carried in the states of cells under the main diagonal.

This can all be done with a domino grammar by passing the symbol information through the east neighbor. We can also easily do it north-west deterministically. We call the symbol information passed from the top row the 'diagonal

signal'. That a signal is being carried simply means that such information is stored on, say, a layer of a cartesian product forming the state set.

On each row, another signal carrying a value from $\{0,1\}$ is sent to the right. This horizontal signal stays 0 until it reaches a cell carrying the diagonal signal. In the cell where the diagonal signal reached it, the horizontal signal value becomes 1. Then, in one step, it becomes 0, and it stays 0 for the rest of the row. That is, the signal carries a 1 on exactly the cells on the main diagonal. Again, by signal we simply mean a bit of information stored on a layer of a cartesian product forming the state set.

On each column, we send yet another signal downward, with values in $\{0,1\}$. This vertical signal is originally 1, and at each cell x it stays 1 if, at x, the diagonal signal is carrying the symbol $p[x]$, for the picture p considered. Otherwise, the signal becomes 0, and it stays 0 for the rest of the column.

Now, we make a state final if and only if both the horizontal and the vertical signal carry a 1 in the state. The horizontal signal makes sure the picture is square, and the vertical one checks that the first row is equal to the reversal of the last column. This concludes the construction.

To show $L_{b=r}$ is not in DREC, consider the set of pictures that are unary everywhere except for the rightmost column and the bottom row, and consider a hypothetical DREC grammar G for this language. Note that during the process of recognizing a picture, the state of the cell in position $(\overline{p}-1,|p|)$ is independent of the contents of the last row of p by how DREC assigns states. Similarly, the state of the cell in position $(\overline{p},|p|-1)$ is independent of the contents of the last column.

By the pigeonhole principle, we can find two words (columns) c^1, c^2 such that $|c^1| = |c^2|$, $c^1_i \neq c^2_i$ for some $1 \leq i < |c^1|$, $c^1_{|c^1|} = c^2_{|c^2|}$, and in the otherwise unary square pictures p^1 and p^2 with these two as the last columns, the same state is assigned to the cells at position $(\overline{p}-1,|p|)$ in both pictures, during the DREC recognition process. Then, consider the pictures q^1 and q^2 obtained by changing the last rows to c^1 in both pictures p^1 and p^2. Due to the claims of the previous paragraph, since p^1 is accepted, also p^2 is accepted. But p^2 is not in L, which is a contradiction. □

Corollary 2.106. *LOC ⫋ DREC.*

Proof. LOC can be easily seen to be closed under rotation, by rotating the patterns counted in pictures. Therefore LOC is a subset of DREC closed under rotation, and thus a proper subset. □

Theorem 2.107. *DREC ⫋ REC.*

Proof. DREC is, by definition, a subset of REC, and therefore by Theorem 2.102, DREC ⊂ DRECR ⫋ REC, since REC is closed under rotations and cartesian products. □

3 Picture walking finite state automata

Recognizability looks at a picture as a whole: proving that a picture p belongs to $L \in \mathrm{REC}$ means simultaneously choosing a state for each position of the picture in such a way that the whole configuration of states is consistent. In the theory of one dimensional formal languages, the class obtained this way coincides with the class of languages obtained using finite state automata that look at one symbol at a time, walking on the cells according to a local rule. As we will shortly see, this is not the case for pictures, with certain natural generalizations of these notions.

3.1 Definitions and basic results

The basic kind of automaton we will be using has a finite memory, and can do both existential (nondeterministic) and universal branching. The following definitions are in no way limited to pictures, but the automata can be run on any canvas. In fact, to run them on pictures, we will first have to decide how to represent the picture as a canvas. Two ways, the bordered and the unbordered representation, were already given in Chapter 2, and we will investigate automata with both representations. Unlike in the case of REC, in most cases changing the representation between these two does not change the associated picture class. However, for certain more restricted representations of pictures, the picture classes obtained are strictly smaller.

It is worth mentioning that, unlike in classical formal language theory, the classes DFA, NFA and AFA usually refer to *two-dimensional four-way* automata and their associated language classes in this thesis, unless otherwise specified. Even in the one-dimensional case, our convention differs from the usual one: by default, one-dimensional DFA, NFA and AFA are *two-directional*, since we use the graph representation inherited from pictures for words. However, we will usually explicitly mention whether a one-dimensional automaton is one-way or two-way.

Definition 3.1. A general h-headed AFA A is a tuple $(Q, U, \Sigma, \Gamma, \delta, I, F)$, where

$$\delta \subset (Q - F) \times (\Sigma \times \{0,1\}^h)^h \xrightarrow{\times} Q \times (\Gamma \cup \{\epsilon\})^h,$$

where $I, F, U \subset Q$ and $\xrightarrow{\times}$ means $\times$, but implies δ is thought of as a multivalued function. We require a universal state to be final if there are no transitions from it, for some input symbol.

The intuition and terminology is

- Q is the set of *states* of the automaton (the finite memory).
- U is the set of *universal states*. When in a universal state, the automaton tries all possible next moves, and accepts if they all accept.
- $Q - U$ is the set of *existential states* or *nondeterministic states*. The automaton accepts if there is a valid next move that leads to an accepting run.
- Γ is the set of *directions*, in a picture the directions might be up, right, down and left, whereas in a binary tree one might be able to go up, and descend to the left and right children.

- δ is the *transition function*, giving the set of *valid moves* based on the current state, and the input symbols and other heads under the heads.
- I is the set of *initial states*.
- F is the set of *final states*.

More formally, let $G = (V, E, r, \Sigma, \Gamma, l, f)$ be a canvas. An *accepting run* of A on G is a finite vertex-labeled rooted tree $T = (V_T, E_T, r_T, Q \times V^h, l_T)$ with root $r_t \in V_T$ and vertices labeled over $Q \times V^h$ by l_T such that

- $l_T(r_T) \in I \times \{r, \ldots, r\}$.
- leaves of T have labels over $F \times V^h$.
- internal nodes of T have labels over $(Q - F) \times V^h$.
- if $x \in V_T$ is an internal node with $l_T(x) = (q, (v_i)_i) \in (Q-U) \times V^h$, then for some $(q', (d_i)_i) \in \delta(q, (l(v_i), H_i)_i)$, x has a child with label $(q', (v_i d_i)_i)$, where H_i has 1 in position j if $v_i = v_j$.
- if $x \in V_T$ is an internal node with label $(q, (v_i)_i) \in U \times V^h$, then for all $(q', (d_i)_i) \in \delta(q, (l(v_i), H_i)_i)$, x has a child with label $(q', (v_i d_i)_i)$ where H_i is as in the case of an existential state.

A accepts G if an accepting run exists.

Definition 3.2. Given a family of labeled canvases $\mathcal{G}$ and an automaton A, we write $\mathcal{L}(A)$ for the subset of $\mathcal{G}$ that A accepts. The family of languages (over a given family of canvases) accepted by h-headed alternating finite state automata is called hHAFA, for any fixed h.

The family of canvases used in each case will be clear from context, and it will usually be one of the canvas representations of pictures.

Informally, we call a state 'rejecting', if there are no transitions from it. We also say that an automaton accepts a picture from state s and head positions $y_1, \ldots, y_h$, if there exists a tree T' where the same local rules apply, but the root has label $(s, (y_i)_i)$ instead. In general, trees where the local properties of a run governing the children of each node hold except for the root and the leaves, are called partial runs. They could be used to more directly formalize the intuition given for the existential and universal states of an hHAFA in the the previous definition.

Note that our automata are always allowed to stay still. In most cases, our automata would not get any weaker in the sense of language recognition by forbidding this, since staying still can be simulated by a deterministic move back and forth. Usually, we will assume all *given* automata have no empty moves, but use empty moves in the automata we construct.

For short, we write AFA for 1HAFA. We will almost exclusively study one-headed AFA. In the definition of an AFA, we drop the coincidence information of heads changing the type of delta to

$$\delta \subset (Q - F) \times \Sigma^h \xrightarrow{\times} Q \times (\Gamma \cup \{\epsilon\})^h.$$

Definition 3.3. An ID of hHAFA A on picture p is a pair $(q, (x_1, \ldots, x_h))$ where q is a state of the automaton, and $\forall i : x_i \in \mathrm{dom}(p)$.

Note that the accepting runs of automata are labeled over ID's.

We only need transition sequences for one-headed automata.

Definition 3.4. Given an AFA A and a canvas G, a *transition sequence* of A on G is a sequence of ID's $((q_i, x_i))_i$ such that

$$\forall i : (q_{i+1}, x_{i+1} - x_i) \in \delta(q_i, l(x_i))$$

and $(q_1, x_1) \in F \times (1, 1)$.

Definition 3.5. An NFA (nondeterministic finite state automaton) is an AFA without any universal states (and U is left out of the definition). A UFA is on without any existential states (U is left out, and taken to be Q). A DFA is an AFA such that δ is a partial function (in which case it doesn't matter whether the states are existential or universal). Note that accepting runs for DFA and NFA are simply transition sequences ending in an accepting state.

The accepting runs of a UFA might be proper trees, and this tree is unique for each input picture, if it exists. If a picture p does not belong to a language L in UFA we will necessarily run into a *proof of exclusion* when trying to build its unique accepting run. By a proof of exclusion, we mean a transition sequence that is either infinite, or cannot be made longer, and does not contain a final state. Since DFA can be though of as UFA, proofs of exclusion also exist for them.

If such a failed run is finite, the last ID (q, v) of this run is such that $\delta(q, l(v)) = \varnothing$. This proves $p \notin L$, because if there is an accepting run, all choices of moves must lead to acceptance. Note that when there is a proof of exclusion, there is also a proof of exclusion that is either finite or eventually periodic by the pigeonhole principle, assuming finite input canvases.

One can also define proofs of exclusion for AFA, but this is slightly more complicated. We will do this in Section 3.3, when such proofs are needed.

Note that UFA and NFA have no obvious connection: Since it is not necessarily possible to make an automaton always halt, there is no clear De Morgan duality. If all automata could be modified into a halting one, one could swap the roles of final states and nonfinal states and show UFA is the class of complements of languages of NFA.

Definition 3.6. An AFA *always halts* on a family of canvases $\mathcal{G}$ if there exist no infinite transition sequences in any of the canvases in $\mathcal{G}$.

Since there are only a finite amount of state-position pairs on finite canvases, it follows that if there are only finitely long sequences like this, there are also only finitely many of them, and their lengths have a common upper bound.

Note that AFA that always halt need not be 'trivial' in any sense, on finite canvases. This is not the case for most natural families of infinite canvases, where such automata must restrict their movement to a finite ball of fixed size, or a 'limit graph' will necessarily exist where the AFA does an infinite search.

Lemma 3.7. *Let L be accepted by an NFA A with k states that always halts. Then, L^c is accepted by a UFA with k universal states that always halts. Symmetrically, for each UFA with k states that always halts, there is an NFA with k states that accepts its complement and always halts.*

Proof. We can use the De Morgan duality, since we don't have to worry about nonhalting: All final states are made nonfinal and vice versa, and existential states are changed into universal states. Now, let $p \in L$. Then the run of A accepting this picture is also a proof of exclusion of the UFA by simply renaming the states. On the other hand, any proof of exclusion for a picture $p \notin L'$ for the language L' of the UFA must be finite by the assumption, and therefore is also an accepting run for p of the automaton A. Therefore $L' = L^c$

Similarly, a UFA can be converted to an NFA. Again, the crucial thing to note is that proofs of exclusion are all finite. □

For a possible example of the lack of duality between these classes, consider Conjecture 3.17. If the statement turns out to be true, UFA would be the only one of the classes DFA, NFA, UFA and AFA not closed under the monotone boolean operations.

We will not distinguish between automata and their classes: XFA denotes both the automata of type X, and the class of languages accepted by finite automata of type X. This should not cause confusion.

In order to run automata on pictures, we have to convert our pictures to canvases. Two ways to do this were already given in Section 2.3: the bordered representation and the unbordered representation.

All canvases corresponding to pictures are road colored with symbols from $\{\triangle, \triangleright, \triangledown, \triangleleft\}$, so we may leave Γ out of the tuples defining the automata. We will, however, talk about certain automata that are not allowed to use all of these directions. This means that after taking the bordered or unbordered representation of the picture, we drop out edges with certain labels, to get a more restricted view of the picture. We define only one such a restricted representations, which can be combined with both bordered and unbordered representations.

Definition 3.8. A *two-way representation* of a picture p is either the bordered or the unbordered representation, with all edges labeled with $\triangleleft$ or $\triangle$ removed from the graph and the road-coloring alphabet, with $\#'$ used as the corner labels of p.

The reason we change the corner cells is that otherwise the automata cannot check whether they are in a corner, unlike in the 4-way case.

In the context of picture languages, for all classes XFA defined so far, we let XFA imply bordered pictures, and FXFA unbordered ones. The classes of picture languages obtained using the two-way representation instead are called 2WXFA and 2WFXFA. In addition to speaking of XFA *automata*, we will speak of FXFA automata. Then, the term FNFA implies to the reader that $\mathcal{L}(A)$ will be taken using the family of unbordered representations of pictures, whereas the bordered representation is used for XFA.

The natural question of connections between these classes immediately arises. We will only consider the connection between XFA and FXFA in this section, and talk more about the restricted versions later. It turns out DFA and NFA have no use for the extra storage space given by the unbordered representation, whereas AFAs do have use for it. The following obvious theorem appears in [8], and we give a proof scetch for it.

Theorem 3.9. *DFA = FDFA on pictures.*

Lemma 3.10. *Let $A = (Q, \Sigma, \delta, I, F)$ be an FDFA. Then there exists a k such that for all $p \in \Sigma_*^*$ there is no accepting computation during which the automaton visits the set $\mathbb{Z}^2 - [-k, \underline{p} + k] \times [-k, |p| + k]$.*

Proof. We prove there is a k such that for all $p \in \Sigma_*^*$ every accepting run stays within $(-\infty, \infty) \times (-\infty, |p| + k]$. From this the claim of the lemma follows by symmetry.

Take $k = |Q|$. Assume the contrary of the claim, and let S be an accepting run on a picture p such that at time step j, the automaton is at (x_1, x_2) with $x_2 > |p| + k$, and let time step i be the last step between 0 and j such that the automaton is on the column $|p| + 1$. The automaton must visit every column in the interval $[|p| + 1, |p| + k + 1]$ between time steps i and j (inclusive).

For each column, choose the last state in which the automaton visits that column in the run S between time steps i and j. Among these $k+1 = |Q|+1$ states, there must be a repetition, so we find two pairs (i_1, c_1) and (i_2, c_2) such that for $m = 1, 2$, at time step i_m, the automaton was in the column c_m, was in the same state q at both time steps i_m, and between time steps i_1 and i_2, it didn't visit the column $|p|$. But then the automaton must have read only #'s during the interval $[i_1, i_2]$, and the run would therefore continue deterministically (and thus periodically) away from the picture, never accepting. □

Using the lemma, the theorem becomes almost trivial.

Proof. Obviously DFA ⊂ FDFA. For the other direction, let $A = (Q, \Sigma, \delta, I, F)$ be a FDFA accepting $L \subset \Sigma_*^*$. We define a DFA A' accepting the same language.

Let k be given by the previous lemma. The DFA A' simulates A while within the picture, and while A is outside the picture, follows it at the edge of the picture, storing how far it is from the border (or a corner). If A goes farther than k, the automaton rejects the picture, since never A can never return by the previous lemma, and periodicity implies it will never accept either. □

A theorem of [9] says NFA = FNFA on pictures *when restricted to words* ($1 \times n$ pictures). The question of whether this is true for pictures of any size was asked in at least [8], [9] and [7]. In Section 3.4, we show this to be the case.

As for AFA, it was also asked in [7] whether AFA = FAFA. We prove a theorem that gives a natural set of pictures in FAFA – AFA by simulating a two-headed AFA on pictures of height 1 using the space above the picture to store the distance between the two heads. This shows that the two classes are different. We need the following well-known fact from the theory of one dimensional languages. A direct proof is given here, since this particular result seems to have surprisingly few easily accessible proofs in the literature.

Lemma 3.11. *Two-way AFA recognize only regular word languages.*

Proof. Given an alternating finite automaton A with states Q, we directly construct a one-way deterministic finite automaton A' with $\mathcal{L}(A) = \mathcal{L}(A')$. We may assume A never tries to walk to the left of the # cell at the left border of the input, and that A never tries to walk to the right of the # cell on the right border. The state set of A' is $\mathcal{M} \times \mathcal{M}^Q$, where $\mathcal{M}$ is the set 2^{2^Q}.

For word w, consider the subset $R = R_\triangleright(w)$ of partial runs of A on w such that if $T = (V_T, E_T, r_T, Q \times V, l_T) \in R$, then $l_T(r_T) \in I \times \{1\}$ and

$$\forall v \in V_T, l_T(v) = (q, i) : \text{leaf}(v) \iff (i = |w| + 1 \vee q \in F),$$

where F is the set of final states of A. That is, R consists of the partial runs that start from an initial state and on top of the leftmost symbol of the input word, and in every branch, either eventually accept, or eventually exit the word from the right.

Somewhat similarly, for word w and state q, consider the subset $R = R_\triangleleft(w, q)$ of partial runs of A on w such that if $T = (V_T, E_T, r_T, Q \times V, l_T) \in R$, then $l_T(r_T) = (q, |w|)$ and

$$\forall v \in T, l(v) = (q, i) : \text{leaf}(v) \iff (i = |w| + 1 \vee q \in F).$$

That is, R consists of the partial runs that start from a given state, on top of the rightmost symbol of the input word, and in every branch, either eventually accept, or eventually exit the word from the right.

For any tree with vertex labels over $Q \times \mathbb{N}$, define $f(T)$ as the set of states not in F appearing in the leaves. Then, after having read the word w (and about to read the symbol at position $|w| + 1$) A' is in state (x, y) where $x = f(R_\triangleright(w)) = \{f(T) \mid T \in R_\triangleright(w)\}$, and

$$\forall q : y(q) = f(R_\triangleleft(w, q)).$$

Since A never walks to the right of the right boundary symbol #, then if we can inductively update this information, we are done: After reading $w\#$, if the automaton A' is in state (x, y), x will only contain subsets of F. Therefore, if x is nonempty, then there must exist an accepting run of A on the input word w, and vice versa.

Let us handle the empty word separately, and assume the automaton is started on some symbol $a \in \Sigma$. A' then initially enters the state $(f(R_\triangleright(a)), (q \mapsto f(R_\triangleleft(a, q)))_q)$. Now consider A' in state (x, y), reading symbol $a \in \Sigma$. For a state $q \in Q$, the trees in $R_\triangleleft(wa, q)$ consist of trees in $R_\triangleleft(w, Q)$ glued together, with partial runs on the singleton word a between them. Therefore, it is clear that knowing $f(R_\triangleleft(w, q'))$ for all $q' \in Q$ determines $f(R_\triangleleft(wa, q))$ for all q, since the behavior of A on a is known. Similarly, it is enough to know $f(R_\triangleright(w))$ and $f(R_\triangleleft(w, q))$ for all q to determine $f(R_\triangleleft(wa))$. □

It should be clear that one-dimensional AFA have exactly the same languages as two-dimensional AFA restricted to pictures in L_{words}, and thus two-dimensional AFA only accept regular subsets of L_{words}, by the previous lemma.

Theorem 3.12. *AFA ⫋ FAFA. More precisely,*

$$\varnothing \neq (\textit{2HAFA} - \textit{AFA}) \cap L_{words} \subset \textit{FAFA} - \textit{AFA}.$$

Proof. Clearly AFA ⊂ FAFA. To prove proper inclusion we will simulate an arbitrary 2-headed AFA on strings, which are represented as $1 \times n$ pictures using the space above the string to remember the distance between the two heads. This proves the claim, because a 2-headed AFA (even a 2HDFA!) can recognize, for instance, the non-regular language $\{a^n b^n \mid n \in \mathbb{N}\}$, while a 2-way AFA restricted to a string will only recognize regular languages by the previous lemma.

More precisely, given a language $L \subseteq \Sigma^*$ recognized by a 2-headed AFA A, we construct a 1-headed FAFA A' recognizing the picture language $L' = \{p \in \Sigma_*^* \mid \overline{p} = 1, p[1, *] \in L\}$, which is not in AFA. We will use $Q \times (S \cup S')$ as the set

of states, where Q are the states of the simulated device and the $S \cup S'$ layer is used for all sorts of bookkeeping. We distinguish one of the states $s \in S$.

Without loss of generality, we may assume that the second head of A is never to the left of the first head, by switching the roles of the heads if this is about to happen. The invariant linking the simulated run of A and the actual run of A' is that if A is in configuration (q, p_1, p_2), where q is the current state of the automaton and p_i is the position of the ith head on the input string (located at $(1, p_i)$ on the input picture), then A will be in the configuration $(q \times s, (1 - (p_2 - p_1), p_1))$. The runs will not be in exact correspondence, but instead, in a run of A', there is a subtree of the shape of a run of A, but where an edge may be replaced by a longer path over $q \times (S - s)$, and nodes may have extra children over $(q \times S')$, starting branches that eventually accept if and only if the simulation is being done correctly.

It suffices to describe how one 'deterministic' step of A is simulated, because we can use a universal or existential state of A' to simulate the same quantification in A (using a path over $q \times (S - s)$). One deterministic step of a computation by A consists of checking if the heads are on top of each other, reading the input symbols at the positions of the two heads, moving them and changing state according to the transition function of A.

To check whether the heads are on the same cell, we check if A' is on the input picture by reading the symbol under its (single) head, checking whether this symbol is a symbol of Σ. To read the symbols under the heads of A, we guess a pair $(a, b) \in \Sigma^2$ using an existential state, and use a universal state to check a is the symbol of the picture to the south of the head of A', and b is the symbol to the southeast of the head, by sending runs over $q \times S'$ checking these properties. The cells in these directions correspond to the positions p_1 and p_2 on the input string of A.

More precisely, when simulating a step, the automaton branches into three in order to read the symbols under the simulated heads. One head, the main branch, respects the invariant, and continues simulating a run of A, whereas the two others branch off, checking that a correct guess about the symbols under the simulated heads was made. These side branches eventually accept if and only if the guess was correct. This means the main branch continues satisfying the invariant if and only if all the side branches eventually accept, and therefore this main branch correctly simulates a run of A.

The moves of the heads of A left and right are easy to translate into local moves of A' in such a way that the invariant is preserved. We must of course detect if one of the heads of A steps outside the input string, which can be done with another (halting) search over $q \times S'$. □

3.2 Closure and nonclosure properties

Theorem 3.13. *All of DFA, NFA, AFA are closed under monotone boolean operations.*

The proof is divided into two cases. We first prove a general lemma about DFA. The basic idea is from [10]. We give a rather complete (read: lengthy) construction and proof.

Lemma 3.14. *For every DFA, there is a DFA with the same language such that the DFA always halts.*

Proof. Let a DFA A over pictures be given. We make a large number of assumptions on A to simplify the construction. For each of these properties, it is clear that for any automaton A'', there is an automaton A with that property, and the same language, without removing any of the other properties. Without loss of generality, we may assume that

- if A accepts, it accepts in the top left corner of the picture.
- A has a unique accepting state f.
- A has a unique initial state i.
- A never reenters i after the initial configuration.
- A, when in state i, first checks it is in the top left corner, and doesn't accept the picture if it isn't.
- whenever A moves onto a #, it always moves back on the next step.
- A has no empty moves.

We construct another DFA A' such that $L = \mathcal{L}(A) = \mathcal{L}(A')$ with the additional property that if $p \notin L$, then the unique proof of exclusion is finite.

First, given a picture p, let us consider the graph G representing all possible computations of A on p with reversed edges. Recall the definition of $\mathrm{edom}(p)$, the union of the domain of p and the border around it. The graph G has the vertices

$$V = Q \times \mathrm{edom}(p)$$

and the edges

$$E = \{((q_2, y), (q_1, x)) \mid (q_2, y - x) \in \delta(q_1, l(x))\},$$

where $y - x$ denotes the direction symbol that would move A from x to y. The graph is obviously finite. The crucial property of this tree needed for this proof is that if $x \in \mathrm{edom}(p)$ and f is the final state, then the weakly connected subgraph H of G containing (f, x) is a tree rooted at (f, x) with all edges pointing away from (f, x).

To prove this, note that in G, all nodes have at most one incoming edge determined by δ. The node (f, x) has no incoming edges because f is a final state, and in particular it cannot be in a cycle. Now consider any simple path u from (f, x) to another node in the undirected version $(V, E \cup E^{-1})$ of G. Because (f, x) has no incoming edges, $u_1 \in E$. All the other edges in u must be in E as well: Assume the contrary. Then, there must be a first edge $u_i \in (E^{-1} - E)$, in which case also $i > 1$. But then, the node between u_i and u_{i-1} has two incoming edges, which is a contradiction.

Now, consider a subgraph C of H that is a simple cycle, and an arbitrary node $c \in C$. By the previous argument, there must exist a directed path u from (f, x) to c using only edges in E. Then there must be an edge in u from some cell outside C to some cell inside C. But this is a contradiction, since all nodes on a cycle must already have an incoming edge from a cell inside the cycle, and all nodes have at most one incoming edge. Therefore H must be a tree.

As was already proven, the edges in H are directed away from (f, x) in the sense that on the unique path in $(V, E \cup E^{-1})$ between (f, x) and an arbitrary

node $(s, y) \in H$, all edges are in E. If $p \in L$, then by the assumption we made on A's acceptance, the pair $(i, (1, 1))$ must be in the tree rooted at $(f, (1, 1))$, for the unique initial state i. Therefore, all we need to do is search the tree H in a depth-first fashion, and accept if the initial state i is seen. Then, it must be in position $(1, 1)$ by the assumption that the automaton checks it is in the top left corner at the beginning of its run.

We will need a total order $(\prec)$ for $Q \times \Gamma$ in the construction. Any fixed total order will do. Let M be the maximal element of this ordered set, and let S be the successor function corresponding to $(\prec)$, with domain $Q \times \Gamma - \{M\}$. The set of states Q' of the new automaton A' constructed for the proof is $(Q \times \Gamma \times \{0, 1\}) \times Q$. We first give the interpretation for the states, and informally explain the behavior of the automaton in each state, and then give the transition rule δ'.

We introduce the concept of prechildren to simplify the proof of correctness. Let $(s, x) \in H$. The prechildren of (s, x) are all (s', xg^{-1}) such that $x, xg^{-1} \in \mathrm{edom}(p)$, and either x or xg^{-1} is in $\mathrm{dom}(p)$. We say that a prechild is proper if it is not a child.

If the automaton A' is in state $((q, g, 1), q')$ in cell x, the interpretation is that the depth-first search through H has just finished searching the subtree of H rooted at (q, xg^{-1}) or returned from a proper prechild, and the search will now continue to the next prechild of (q', x), as dictated by S, if such prechildren are left. If none are left, the automaton will move to the parent of (q', x) in H by applying the transition function.

If the automaton A' is in state $((q, g, 0), q')$ in cell x, the interpretation is that it is about to start searching the subtree rooted at (q', x), a prechild of the configuration (q, xg). However, (q', x) may be a proper prechild, in which case $((q, xg), (q', x))$ is not an edge of H. In this case, the search is cut, and the automaton returns to configuration (q, xg).

Note that in both of the previous cases, the direction g stored in the state is towards the root of H.

The automaton A' will always know whether a move in some direction $g^{-1} \in \Gamma$ moves to a prechild, since the automaton knows whether the symbol under it is a $\#$, it knows which direction it came from, and A cannot directly move from one border cell to another by the assumptions. We use this information in the definition of the transition rule δ'.

First, let the automaton A' be in state $((q, g, 0), q')$, and let the symbol under its head be $a \in \Sigma$. If $q' \neq f$ and we are in the case of a proper prechild, that is, $\delta(q', a) \neq (q, g)$, then δ' is given by

$$\delta'(((q, g, 0), q'), a) = (((q', g, 1), q), g).$$

If, however, $q' = f$ or $\delta(q', a) = (q, g)$, then if $q' \in I$, the automaton accepts, and otherwise

$$\delta'(((q, g, 0), q'), a) = (((q', g_1, 0), q_1), g_1^{-1})$$

where (q_1, g_1) is the first prechild of (q', x).

Now let the automaton be in state $((q, g, 1), q')$, and again let the symbol under its head be $a \in \Sigma$. If (q'', g'') is the first element of $S((q, g)), S^2((q, g)), \ldots$ such that (q'', xg''^{-1}) is a prechild of (q', x), then

$$\delta'(((q, g, 1), q'), a) = (((q', g'', 0), q''), g''^{-1}).$$

If, however, there are no prechildren left, then if $\delta(q', a) = (q'', g'')$, then

$$\delta'(((q, g, 1), q'), a) = (((q', g'', 1), q''), g'').$$

If $\delta(q', a)$ is not defined, then we simply leave δ' undefined. This includes the case $q' = f$.

We will prove, for any node $(q', x) \in H$, by induction on the size of the subtree rooted at (q', x), that if A' is started in the cell at x in state $((q, g, 0), q')$ such that either (q', x) is a child of (q, xg) in H, or q' is a final state, it will accept p if there is an initial node in that subtree, and will otherwise eventually return to the parent (q, xg) of (q', x) if $q' \notin F$, and reject otherwise.

First, assume (q', x) is a leaf, let $l(x) = a$, and consider A' in cell x in state $((q, g, 0), q')$. By the assumption, $\delta(q', a) = (q, g)$. Since the subtree rooted at (q', x) is a singleton, it contains an initial node if and only if q' is an initial state if an only if the automaton accepts in state $((q, g, 0), q')$, as required.

Otherwise,

$$\delta'(((q, g, 0), q'), a) = (((q', g_1, 0), q_1), g_1^{-1})$$

where (q_1, g_1) is the first prechild of (q', x). Since, by the assumption that (q', x) is a leaf, (q_1, xg_1^{-1}) is a proper prechild of (q', x). Therefore, the automaton will instantly return to cell x in state $((q_1, g_1, 1), q')$. This will happen for all the prechildren in order, until finally the automaton ends up in cell x in some state $((q_k, g_k, 1), q')$ such that there are no valid prechildren left. Then, if $\delta(q', a) = (q'', g'')$, the automaton returns to cell xg'' in state $((q', g'', 1), q'')$. But by the assumption, $\delta(q', a) = (q, g)$, which concludes the proof for the base case.

As for the general case, consider the subtree rooted at configuration (q', x), and consider A' in cell x, in configuration $((q, g, 0), q')$ such that $\delta(q', l(x)) = (q, g)$. In this case, the automaton will behave just like in the base case, and enter all prechildren in succession, since by the inductive assumption, the automaton returns exactly the same way from a proper prechild as it does from a child, and therefore it also returns to the parent of (q', x) correctly.

We now just need to note that if the subtree rooted in (q', x) contains an initial node, then this node is either (q', x), or it is in one of the subtrees. In the first case, after checking (q', x) is actually a child of its parent, it will check whether (q', x) is an initial node, and accept p if it is. In the second case, it follows by induction that A' accepts p after entering this child.

Now, the result follows, since there is an initial node in H if and only if there is an accepting computation of A from the final state in the top left cell, by the assumptions we made on A. The unique proof of exclusion is finite because H is finite. □

Lemma 3.15. *DFA are closed under all boolean operations.*

Proof. Let A_1 and A_2 be DFA's with languages L_1 and L_2, respectively. We explain the construction of automata for L_1^c, $L_1 \cup L_2$ and $L_1 \cap L_2$.

For L_1^c, first construct an automaton A_1' that always halts, with $\mathcal{L}(A_1') = L_1$. Then, consider the automaton B obtained from A_1' by changing final states to rejecting states, and rejecting states to final states. On pictures $p \in L_1$, the automaton B will the exactly simulate the unique run of A_1' on p. This run will eventually reach a final state of A_1', from which there are no transitions.

Therefore B will reject p. Conversely, if $p \notin L_1$, the run of A_1' being simulated eventually halts without accepting, that is, it reaches a rejecting state. But such states are final in B.

For $L_1 \cup L_2$, we take a halting automaton A_1' for L_1, and construct an automaton B that works on picture p as follows: First, a run of A_1' on p is simulated until A_1' would have reached a final or a rejecting state. If A_1' would've accepted, B accepts as well. This takes care of pictures $p \in L_1$. If, however, A_1' would've rejected p, then B moves to the top left corner, and simulates a run of A_2, accepting if and only if A_2 accepts. This takes care of pictures $p \in L_2$. The crucial thing to note is that the automaton A_1' always halts, so B always eventually checks inclusion of p in L_2 if $p \notin L_1$.

For intersection, B works similarly, inclusion is first checked in L_1, then in L_2, accepting if and only if both automata simulated accepted p. In this case, A_1 does not need to be made to always halt, since if it never halts, p is not in L_1, and thus $p \notin L_1 \cap L_2$. □

Lemma 3.16. *NFA and AFA are closed under monotone boolean operations, and UFA is closed under intersection.*

Proof. Let A_1 and A_2 be XFA's for L_1 and L_2, respectively.

Since NFA and AFA are allowed to use existential states, inclusion of given picture p in the union of L_1 and L_2 is trivial to check: the automaton simply guesses which language L_i the picture belongs to, and simulates the corresponding automaton A_i.

For intersection, the construction we used for DFA still works. For any XFA, let A_1 and A_2 be automata of type XFA. When an accepting state of A_1 is reached, the automaton deterministically moves to the top left corner of p, and starts a run of A_2. The accepting runs of such an automaton are runs with an accepting run of A_1, with accepting runs of A_2 attached to the leaves, separated by a chain of length less than $|\mathrm{dom}(p)|$. □

We can also easily see that the classes are closed under cartesian products: Just like in the proof of closure under intersection: One layer at a time, it is checked that that layer of the picture belongs to the corresponding language. Since all layers must belong to their corresponding languages, it follows that if the picture is in the cartesian product, all these checks eventually halt.

As an example of a possible asymmetry between NFA and UFA, we give the following conjecture.

Conjecture 3.17. *UFA is not closed under union.*

An intuitive idea of why this might not be the case is given by the following argument: Consider the language

$$(L_{acyclic} \| \Sigma_*^*) \cup (\Sigma_*^* \| L_{acyclic})$$

where $\|$ denotes some kind of separating border between the two languages. Both these languages are individually in UFA, but for neither language does there exist a UFA that always halts by Corollary 3.47, that is, some graph will always make it loop forever, as long as it stays on one side of the picture. The intuitive leap is that there can be no essentially useful information exchange between the two sides (which seems probable, since either side can, individually,

contain absolutely anything). If this is true, the UFA would have to try the sides one by one to find the side on which no loops occur, but this is impossible to do.

Similar argumentation could be made for the language $(L_{acyclic} \times L_{graphs}) \cup (L_{graphs} \times L_{acyclic})$.

Not being closed under union, some might argue, could also serve as a reason not to further investigate this class, since this is one of the most frequent naturally occurring closure properties. Of course, even though most of the classes have many of the 'categorical' closure properties, all the automata classes lack many 'positional' closure properties due to their local nature, which makes UFA stand out slightly less.

In particular, we show they are not closed under concatenation. We assume this is due to the fact automata with only a finite amount of memory cannot remember where they are, and thus cannot know where a picture from one language changes into a picture from another one, and might parse the picture differently in their finite memory every time they pass this border. Of course, this kind of argumentation can not (yet!) be made mathematically precise in general, so we reduce the claim to the pigeonhole principle instead. We will need Theorem 3.43 from the next section in the proof. The proof of Theorem 3.43 is independent of the considerations of this chapter.

Theorem 3.18. *None of the classes XFA are closed under concatenation.*

Proof. Consider the language $L_{l=r} = \{p \in \Sigma_*^* \mid p[*,1] = p[*,|p|]\}$, where $|\Sigma| > 2$. Clearly, $L_{l=r}$ is in DFA, and therefore in all of the classes. Consider the language $L = L_{l=r}^{3\oplus}$. If one of the classes XFA were closed under concatenation, then L would be in this class. We show that L is not even in the class AFA. If it were, then L^c would be in co-AFA, and therefore also in REC by Theorem 3.43. We will show, however, that L^c is outside REC, proving the claim. Therefore, let us suppose, on the contrary, that L^c were in REC.

L^c is the language of pictures p such that

$$\begin{gathered}\forall i,j \text{ such that } 3 \le i < j \le |p| - 2 : \\ p[*,i] = p[*,j] \implies p[*,1] \neq p[*,i-1] \vee p[*,j+1] \neq p[*,|p|]\end{gathered}$$

Consider three distinct symbols $a, b, c \in \Sigma$. Then also the intersection L' of L^c with the (clearly recognizable) language

$$L'_{stripes} = c^{*\ominus} \oplus (c^{*\ominus} \oplus \{a,b\}^{*\ominus})^{*\oplus} \oplus c^{*\ominus} \oplus c^{*\ominus}$$

would be in REC.

This language is the subset of $L'_{stripes}$ such that none of the columns over $\{a,b\}$ are equivalent: Such a picture is in L' since $p[*,i] = p[*,j]$, for $3 \le i < j \le |p| - 2$ can then only be true for two columns over c, but then the columns next to i and j would be over $\{a,b\}$, and therefore distinct from the first and last columns. On the other hand, if in a picture $p \in L'_{stripes}$, two columns i and j of p over $\{a,b\}$ were equivalent, then $p \notin L^c$, because the columns just outside i and j would be over $\{c\}$, and therefore equal to the leftmost and rightmost columns, respectively.

Finally, we obtain that that if L' were in REC, then also

$$L_{c \neq c} = \{p \mid \nexists\, i \neq j : p[*,i] = p[*,j]\}$$

would be in REC because REC is closed under removing the first and last column, and every second column by Corollary 2.44 and Corollary 2.46.

But $L_{c \neq c}$ was proven non-recognizable in Section 2.3. □

For 2WNFA, we can find a nice characterization, which gives a rather complete picture of what closure properties this class has. Its corollary will also be used in Section 3.4. To prove the characterization, we will need the definition of a semilinear set, and the following well-known theorem, which we state without proof. A proof of Theorem 3.21 can be found in [19].

Definition 3.19. Let Σ be ordered as $a_1, \ldots, a_n$. Then, we define the Parikh vector of $w \in \Sigma^*$ as $(|w|_{a_1}, \ldots, |w|_{a_n})$, where $|w|_a$ gives the number of occurrences of a in w. To every word language $L \subset \Sigma^*$, we associate a language of Parikh vectors in the obvious way.

Definition 3.20. A linear set is a set of the form

$$x + \langle x_1, \ldots, x_k \rangle$$

where $x, x_1, \ldots, x_k$ are vectors of $\mathbb{Z}^n$ and $\langle X \rangle$ denotes the submonoid of $\mathbb{Z}^n$ generated by X. A semilinear set is a finite union of linear sets (all with the same vector length n).

Clearly the linear subsets of $\mathbb{N}^2$ are exactly

$$\{x + \langle x_1, \ldots, x_k \rangle \mid x \in \mathbb{N}^2, x_1, \ldots, x_k \in (\mathbb{N} \cup \{0\})^2\}.$$

Theorem 3.21. *The Parikh languages of context-free and regular languages are semilinear sets.*

Definition 3.22. We define SL as the picture class of unary languages L such that shape(L) is a semilinear set.

The following theorem contains one of the key ideas of Section 3.4.

Definition 3.23. For a transition sequence $(s_i)_i = (q_i, x_i)_i$, we write seq$((s_i)_i)$ for the sequence of moves used in $(s_i)_i$.

Theorem 3.24. $\mathit{2WNFA} \cap 1^*_* = \mathit{SL} \cap 1^*_*$.

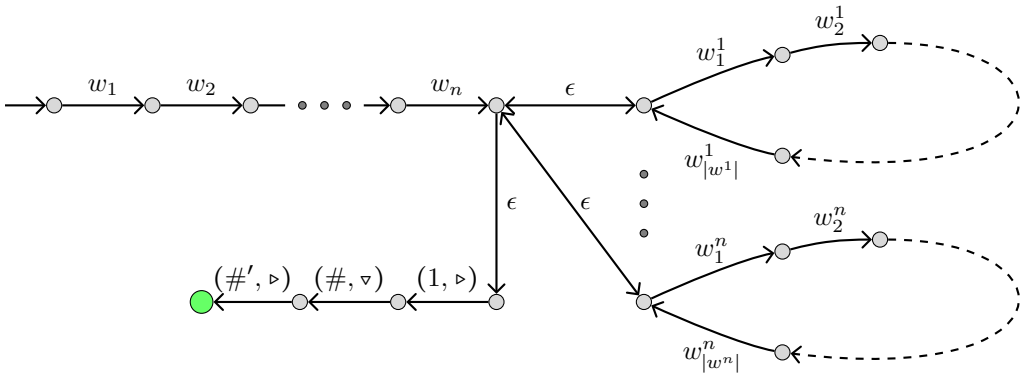
Proof. Let A be a 2WNFA. We may assume A only accepts on top of $\#'$, since we can simply make the automaton move to $\#'$ when it would've accepted, and accept only then. We will prove A recognizes a semilinear set. To prove the claim, we construct a one-dimensional NFA A' such that

$$L = \mathcal{L}(A') = \{w \in \Gamma^* \mid \exists p \in \Sigma^*_* : \exists \text{ accepting run } r \text{ of } A \text{ on } p \text{ with seq}(r) = w\}$$

where Γ is the set $\{\triangleright, \triangledown\}$ of possible moves on the graphs representing pictures.

Such an automaton is easy to construct: A' simulates the finite state machine of A, and remembers whether the head of A is in the domain or one of the borders of a hypothetical picture p. A' first feeds 1 to the transition function of A. It then simply guesses if and when a border is entered, and starts feeding # to the automaton. If A' reads $d \in \{\triangleright, \triangledown\}$, it always chooses a move of A that would have moved it in that direction. Note that all A' has to remember about the

Figure 5: A finite state automaton with language L such that $\text{shape}(L) = S_j$.



hypothetical picture p is whether a border has been reached, and which border it is, since a 2WNFA can never exit a border once it has been entered.

Now, the language L' of Parikh vectors associated with L is a semilinear set. But clearly $L' = \text{shape}(\mathcal{L}(A))$. Therefore, $\text{2WNFA} \cap 1_*^* \subset \text{SL} \cap 1_*^*$.

Now, consider an arbitrary semilinear set $S = S_1 \cup \ldots \cup S_n$, and the associated picture language $L \subset 1_*^*$. Since our 2WNFA can guess which one of the S_j a given picture p is in, it is enough to show how to to accept one of the linear sets $S_j = x + \langle x_1, \ldots, x_k \rangle$. So, let us construct a 2WNFA A with language $L \subset 1_*^*, \text{shape}(L) = S_j$. Let $x = (a, b), x_i = (a_i, b_i)$, and let $w = {\triangledown}^{a-1}{\triangleright}^{b-1}, w^i = {\triangledown}^{a_i}{\triangleright}^{b_i}$.

The automaton depicted in Figure 5 has the required language. A transition with label $d \in \{\triangleright, \triangledown\}$ denotes a transition in direction d, reading 1. A transition with label (a, d) denotes a transition in direction d, reading a. The transition graph consists of an initial segment, a set of loops, and a final segment. The initial segment moves the automaton to position x, each of the loops i is a move by exactly the vector x_i, and the final segment checks the automaton is exactly in the bottomright corner of the picture. □

Corollary 3.25. *SL ⊂ NFA.*

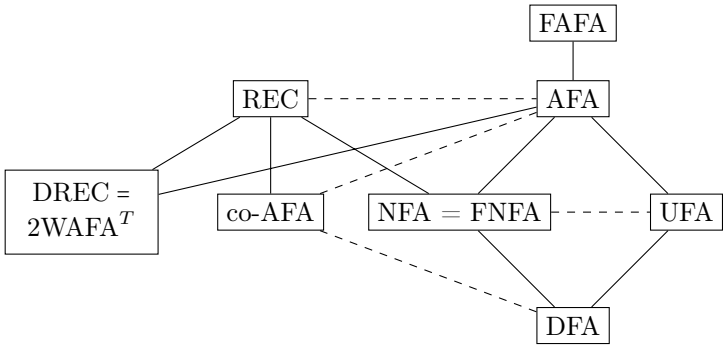
In the next section, we will strengthen the previous corollary by finding a unary language in NFA – SL.

3.3 Connections between classes

In this section we show some connections between the classes defined in this chapter, and the classes of chapter 2. We will not compare all pairs of classes defined sofar. The most interesting results are summarized in Figure 6. A solid line from A to B, when A is on top, means A is a proper superset of B. A dashed line denotes the incomparability of two classes.

It is clear that $\text{DFA} \subset \text{NFA}, \text{UFA} \subset \text{AFA}$. The first sequence of theorems in this section shows that these inclusions are proper, and that NFA and UFA are incomparable. The results concerning UFA have not been mentioned in the

Figure 6: Hasse diagram of automata classes. Some uninteresting relations are omitted.



literature as far as we know, otherwise this chapter mostly follows [7]. The interesting result that 2WAFA^T = DREC is from [11]. A less formal proof is given here.

The theorems are based on one-dimentional automata running on unary inputs. Such automata can be thought of as accepting the unary representations of a set of numbers. Because of this, we sometimes abbreviate $1^n \in L$ as $n \in L$.

The following lemma is found in the article [7] apart from the result concerning UFA, although all the ideas are already explicit in their proof. The lemma is of individual interest, and shows an interesting symmetry between automata defined by existential and universal quantification in the 1D case. All the noninclusions between the four basic XFA classes in Figure 6 are based on this lemma, and suitable choices of separating languages.

Our statement of the lemma turns out slightly nicer than that of [7] due to the fact we may write k instead of $k+2$ as the threshold of the pumping lemma. This is because the automata of [7] consider the leftmost and rightmost cells special, whereas our automata will have to sense borders programmatically, by going outside the input and coming back, since our definition of one dimensional automata is based on the general one specifically tailored to also accommodate the FXFA classes. (On the other hand, we are less lucky in the statement of Lemma 3.30, where our statement uses $h+2$ instead of the h of [7].)

Lemma 3.26. *Let A be a 1D (two-way) AFA with k states over a unary alphabet $L \subset 1^*$. Let $n > k$. Then*

- *if A is an NFA, $1^n \in L \implies 1^{n+k!} \in L$.*
- *if A is a UFA, $1^n \in L \impliedby 1^{n+k!} \in L$.*

In particular, if A is a DFA, an equivalence holds, since if the transition relation is a function, we may change between universal and existential states without changing the language.

Proof. First, assume all states are existential. Then, if $1^n \in L$, we may assume there is an accepting run of the NFA that starts and ends at the left end of the input. All runs of the NFA can be split into partial runs from the end

markers # to another end marker # (possibly the same one). Consider such a partial run $s_1, \ldots, s_m$ of A from the left end marker # of the input to the right end marker # of the input. That is, a partial run such that the automaton only visits the left and right end markers at the first and last step of (s_i). Let $s_1 = (q, x), s_m = (q', x')$. We show how to make a similar partial run for the input $1^{n+k!}$ from the left border to the right border, from state q to state q', that only enters border cells at the beginning and the end of the run.

For this, note that by the pigeonhole principle, there must exist $1 < i < j < m$ such that $s_i = (q, p), s_j = (q, p'), 1 \leq p' - p \leq k$ for some $p < p'$. This is because the automaton must go through at least n, and therefore more than k symbols of the input, and we can choose a representative state for each of these cells, when they are first seen by the automaton. Because $k!$ is necessarily divisible by $p' - p$, we may repeat ('pump') this part of the computation to obtain a partial run with the desired properties for the input $1^{n+k!}$, since the automaton will not visit the borders of the input during either partial run.

Consider now a partial run from the left marker back to itself. It is clear that the same partial run works in the long input. Symmetrically, we can pump a partial run from right to left to obtain partial runs for the long input from partial runs for the short input, and partial runs from the right end marker back to itself can directly be used as partial runs for the long input. Now, because for each partial run for the short input, there exists a partial run for the longer input between the same end markers, we may do this pumping for every partial run, to obtain an accepting run for the longer input.

Assume now that all states are universal. Then, we consider proofs of exclusion instead. That is, we assume $1^n \notin L$, and prove $1^{n+k!} \notin L$. The simple case is when there is a finite proof of exclusion. Then the exact same construction as in the NFA case can be used to find a finite proof of exclusion for the long input. However, it is possible that all proofs are infinite. Consider such a proof $(s_i)_{i \in \mathbb{N}}$. Again, we split this run into partial runs. If all these runs are finite, we obtain another infinite sequence by changing all the partial runs to work in the longer input. Otherwise, there exists a last time j when the automaton visits an end marker in (s_i). Clearly, the infinite partial run after time j will not touch any of the end markers in the longer input either. Therefore, $1^{n+k!} \notin L$. □

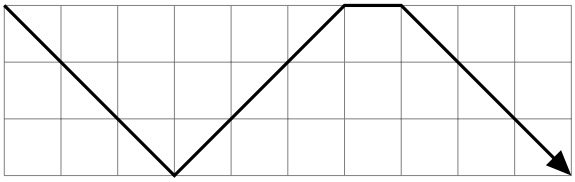
Definition 3.27. The billiard ball language $L_{billiard}$ is defined as

$$L_{billiard} = \{p \in \{1\}_*^* \mid |p| \in \langle \underline{p}, \overline{p} + 1 \rangle\} = \{m\overline{p} + n(\underline{p} + 1) \mid m, n \in (\mathbb{N} \cup 0)\}$$

Let us consider $L_{billiard}$ from an automata theoretic standpoint. More intuitively, the billiard ball language can be defined as the unary language defined by a billiard ball moving according to the following set of rules along *the corners* of the cells of the input picture:

- The billiard ball starts at the top left corner of the input.
- The ball moves diagonally over the input, from left to right.
- When the ball hits the top wall, it bounces southeast.
- When the ball hits the bottom wall, it bounces northeast.
- When the ball bounces from a wall, it can choose to slide to the next corner on the right.

Figure 7: An accepting run for the unary 3x10 picture, which is in $L_{billiard}$.



- The ball accepts the input if it can reach a right corner of the input.

These movement rules can almost directly be translated into an NFA, by just keeping track of the corner of the current cell where the billiard ball is. This gives us the following.

Lemma 3.28. *$L_{billiard} \in$ NFA, and there even exists an always halting NFA for it.*

See Figure 7 for an example of the movement of the ball in an accepting computation of such an NFA.

Let us characterize the numbers that occur as widths in the pictures of $L_{billiard}$, for a fixed h.

Lemma 3.29. *Let $W(h) = \{w \mid (h, w) \in shape(L_{billiard})\}$. Then, if $w = ah + b$ for some $a \geq 0, 0 \geq b < h$, then*

$$w \in W(h) \iff a \geq h \vee b \in \{0, ..., a\}$$

Proof. If $a \geq h$, then $w = (a-b)h + b(h+1)$, where $a - b > 0$. For other w, consider $Z = W(h) \cap [ah, (a+1)h - 1]$. It is clear that $[ah, ah + a] \subset Z$, since $[ah, ah + a] = \{ah, (a-1)h + (h+1), (a-2)h + 2(h+1), \ldots, a(h+1)\}$.

Now, consider $xh + y(h+1) \in Z, x, y \geq 0$. We must have $x + y \leq a$ since $x + y \geq a + 1$ implies

$$(a+1)h - 1 \geq xh + y(h+1) \geq (x+y)h \geq (a+1)h \implies -1 \geq 0,$$

which is a contradiction.

On the other hand, we must have $x + y \geq a$ since $x + y \leq a - 1$ implies

$$xh + y(h+1) \leq (x+y)(h+1) \leq (a-1)(h+1) = ah + (a-h) - 1 < ah.$$

But as we saw, $x, y \geq 0, x+y = a$ gives us exactly $[ah, ah+a]$, which concludes the proof. □

The usefulness of the language $L_{billiard}$ comes from the fact that for a fixed h, the sequence of allowed widths is cofinite, but becomes periodic rather late, since $h^2 - h - 1 = h(h-2) + (h-1) \notin L_{billiard}$ by the previous lemma. This fact, the pumping lemma given earlier, and the following lemma are enough to prove $L_{billiard} \notin$ UFA, and in particular $L_{billiard} \notin$ DFA.

Lemma 3.30. *Let A be an AFA with k_u universal states and k_e existential states recognizing the unary picture language $L \subset \{1\}^*_*$. Then, for each h, the language $L_h = \{1^{|p|} \mid p \in L, \underline{p} = h\}$ is recognized by a one-dimensional two-way AFA with $k_u(h+2)$ universal states and $k_e(h+2)$ existential states.*

Proof. Let h be given, and $\Sigma = \{1\}$. We construct a one dimensional AFA A' recognizing the language L_h with the required number of states. This set Q' of states is

$$\{(q,i) \mid q \in Q, 0 \leq i \leq h+1\}.$$

The i part simulates the height of the automaton A on the input picture, while q simulates its current state. State (q,i) is universal or final if and only if q is universal or final in A, respectively. The initial states of A' are $(q,1)$ for initial states q of A.

The idea is, as always, to translate transitions of A to those of A' in such a way that accepting runs of A for a picture exist if and only if accepting runs exist for A' on the word $w = 1^{|p|}$ corresponding to p. We do this by ensuring that the transition from $(q,(i,j))$ to $(q',(i',j'))$ is valid for A on p if and only if the transition from $((q,i),j)$ to $((q',i'),j')$ is valid for A' on w.

Therefore, the bijection

$$f((q,(i,j))) = ((q,i),j)$$

between ID's of A and A' can be used to turn accepting runs of A on p into accepting runs of A' on w. This is true because initiality, finality, existentiality and universality of nodes are preserved under f, and $c_1 \to c_2 \iff f(c_1) \to f(c_2)$ for configurations c_1, c_2 of A. Also the converse is true, using the bijection f^{-1}.

Now consider A in configuration $(q,(i,j))$. It reads 1 if and only if

$$1 \leq i \leq h \wedge 1 \leq j \leq |p|.$$

The automaton A' can deduce whether this is the case from its state, and the symbol it reads: If $i \in \{0, h+1\}$, then the state (q,i) of A' contains this information. If $i \in \{0, |p|+1\}$, then A' is on the border of w, and thus A' reads $\#$ as well. Also, A' can read the state of A directly from its own state.

Therefore, we know how A' can deduce the possible transitions that are valid for A. These are easy to map into moves of A': A change of state of A from q to q' is simulated by changing the left side of Q'. Moves up or down by A are simulated by changing the right side of Q', and moves left or right are simulated by moving the automaton A' left or right, leaving the right side of its state unchanged. This concludes the construction of A' having the required properties. □

We are now ready to prove all the noninclusions between the automata classes.

Theorem 3.31. *UFA ⋈ NFA.*

Proof. Assume A is a UFA with k states recognizing $L = L_{billiard}$. By the previous lemma, for each h, the language L_h is accepted by a UFA with $k(h+2)$ states. If we let h be big enough that $h^2 - h - 1 > k(h+2)$, Lemma 3.26 tells us that

$$\forall n \in \mathbb{N} : h^2 - h - 1 + n(k(h+2))! \notin L_h.$$

But we know that L_h is cofinite, which gives a contradiction. Therefore there cannot be such an A, from which it follows that NFA $\not\subset$ UFA.

For the other direction, assume on the contrary that UFA $\subset$ NFA. Then because there exists an always halting NFA for L, there exists an always halting

UFA for its complement L^c by Lemma 3.7. Since UFA $\subset$ NFA, this means $L^c \in$ NFA. But L^c is not in NFA by an argument similar to that of the previous paragraph: Assume A is an NFA with k states recognizing L^c. By the previous lemma, for each h, the language L_h^c is accepted by an NFA with $k(h+2)$ states. Let $h^2 - h - 1 > k(h+2)$. Then,

$$\begin{aligned} h^2 - h - 1 \notin L_h \quad &\Longrightarrow\ h^2 - h - 1 \in L_h^c \\ &\Longrightarrow\ \forall n \in \mathbb{N} : h^2 - h - 1 + n(k(h+2))! \in L_h^c \\ &\Longrightarrow\ \forall n \in \mathbb{N} : h^2 - h - 1 + n(k(h+2))! \notin L_h, \end{aligned}$$

which means L_h is not cofinite, a contradiction. □

Directly from the proof of Theorem 3.31 we have nonclosure under complementation for both of the classes.

Corollary 3.32. *Neither of NFA and UFA is closed under complementation.*

Theorem 3.33.

$$DFA \subsetneq NFA \subsetneq AFA$$

and

$$DFA \subsetneq UFA \subsetneq AFA.$$

Proof. All that needs to be proved is that all the inclusions are proper. For this, note that if NFA were the same class as DFA, UFA would be a superclass of NFA, which is impossible, and therefore NFA is a proper superclass of DFA. Similarly, we see that UFA is a proper superclass of DFA. Since AFA is a superclass of both NFA and UFA, it must be a proper superclass of both for the same reason. □

As another application of $L_{billiard}$, let us prove NFA is a proper superset of SL even in unary.

Theorem 3.34. $SL \cap 1_*^* \subsetneq NFA \cap 1_*^*$

Proof. Let X be the shape($L_{billiard}$). By Corollary 3.25, all we need to prove is that X is not semilinear. So let us assume, on the contrary, that X it is semilinear. We may ignore the tuples $(h, 0)$ since there is only one empty picture. We define two diagonal subsets $D_1\{(h,h) \mid h \in \mathbb{N}\}$ and $D_2 = \{(h, h+1) \mid h \in \mathbb{N}\}$ of X. Now consider an arbitrary linear subset $S = x + \langle x_1, \ldots, x_k \rangle$ of X. We think of the rows of X as those of the lower right quarterplane.

Since X contains no element under the cells of D_1, it is clear that the x_i cannot contain a vector with angle more than $\pi/4$ clockwise from the horizontal axis. Therefore, every cell in D_1 must be generated by vectors with exactly this angle. Also, vectors of D_2 far enough from the origin will use at most one vector of angle less than $\pi/4$: Using even one will move them out of D_1, and using another will necessarily move them out of D_2 as well.

Now consider a vector $v \in D_2$ such that $v = x + \sum_{1 \le i \le k} a_i x_i$, where say $a_1 = 1$ and x_1 is the unique vector with angle less than $\pi/4$. By taking a large enough v, we can also guarantee $\exists i \neq 1 : a_i \neq 0, x_i \neq (0,0)$ such that the angle of x_i is $\pi/4$. Now clearly the set $\{x + 2x_1 + mx_i \mid m \in \mathbb{N}\}$ contains vectors not in X, since the distance from a cell of D_2 to the next element of X on the right increases without bound. □

We now proceed to compare automata classes with REC and DREC. The constructions given here are also from [7].

It turns out the class NFA is properly contained in REC by a rather straightforward construction that draws the run of an NFA on the picture, and checks its local consistency. The only problem is getting rid of cycles: We cannot simply tell the grammar to start with an initial state at the topleft corner, and add states as the NFA moves on the picture; the grammar might end the run from the initial state in a cycle, and then add another run starting from a random position on the picture, ending in a final state. However, since the accepting runs of an NFA are sequences, this is easily avoided by having unique next transitions from every position: then the only way to have start a sequence that ends in a final state is to start it from the unique initial state.

Theorem 3.35. *NFA* $\subset$ *REC.*

Proof. Let A be an arbitrary NFA with states Q, with the additional properties that A works deterministically on top of #, and no initial state is ever entered during a run after the first transition. We explain how to construct a recognizable grammar G for $L = \mathcal{L}(A)$. The states Δ of G store the current symbol and a set $S \subset 2^Q$ of states a hypothetical NFA run might've visited in the current cell. For all states $q \in S$ in the state of cell x, such that $q \notin F$, it is also stored which transition in $\delta(q, l(x))$ a hypothetical NFA run takes. A state $q \in Q - F$ is not allowed in any state of Δ if it does not have outgoing transitions.

The left corner cell must contain an initial state. Also, there must be a final state in the state of exactly one cell. This is easy check, for instance by using a locally threshold testable grammar instead of a local grammar.

Let cell x have the state (a, S, h), and let $q \in S - F$. Let $(q', g) \in \delta(q, l(x))$ be the transition taken from state q in this cell (as determined by h). If the cell xg is in $\mathrm{dom}(p)$, then it is checked, by the local rule, that the cell xg has q' in its subset of states. If, however, xg is outside of $\mathrm{dom}(p)$, then we require the unique state in which A returns to x to be in S.

It is also checked that these are the only ways in which a state q could appear in S. That is, for every $q \in S$ it is checked that one of the neighbors of x, or x itself, adds q to S due to a transition stored in the state.

Now, consider a picture $p \in L$, and an accepting run r of A for it. If r contains a loop, that is $r_i = (q_i, x_i) = (q_j, x_j) = r_j$ for some $i < j$, then we may remove this loop, to obtain a shorter accepting run. Therefore, there will always exist a loopless accepting run of A on p. For each $r_i = (q_i, x_i)$ such that $x_i \in \mathrm{dom}(p)$, we add q_i to the state of cell x_i, and have the outgoing transition be the one taken from r_i. If $x_i \notin \mathrm{dom}(p)$, then x_{i+1} is on $\mathrm{dom}(p)$, and we simply ignore r_i completely. Because there are no loops, all outgoing transitions are determined exactly once. This clearly gives a picture in $\mathcal{L}(G)$.

Consider the other direction. Let $p \in \mathcal{L}(G)$ with assignment of states p'. Using p', we construct an accepting run r of A on p. Consider the unique accepting state f on some cell x of p'. Let $r_1^{-1} = (f, x)$. By the local rule, this accepting state must have been added onto x by the local rule, based on an outgoing transition from some neighboring cell, or x itself. We choose an arbitrary one of these. If the forcing cell is a neighbor, we append the forcing state–cell pair (q, y) to r. If the forcing cell is x itself, there must exist a partial run of length 2 through # that ends with f being forced on x. We append that partial run to r^{-1}, reversed.

The process of adding such pairs to r^{-1} is repeated until an initial state is reached. This must eventually happen, since a pair cannot be repeated in the construction of r^{-1}: Consider the first repeated pair (q, y). Clearly q cannot be final This pair has a unique outgoing transition, so it could only have forced whatever preceded it when it was added to r^{-1} for the first time. But this means the cell it forced must have been repeated already.

But, when an initial state is reached, we have, in fact, constructed a reversed accepting run of A, and therefore $p \in L$. □

REC itself is not closed under complementation, but it turns out the complements of languages in NFA are always in REC. In fact, the complement of every language in AFA turns out to be recognizable. Before proving this, let us formalize the idea of a complement class, and discuss it briefly.

Definition 3.36. Let CLS be a picture class. Then the set *co-CLS* contains exactly the languages

$$\{\Sigma_*^* - L \mid \Sigma \text{ is an arbitrary alphabet}, L \in \text{CLS}\}.$$

Lemma 3.37. *If CLS is a picture class, then co-CLS is a picture class.*

Proof. We just need to check that co-CLS is closed under bijective letter substitutions. Let $L \in$ CLS with $\text{alph}(L) = \Sigma$, let Σ' be an arbitrary alphabet, and let $f : \Sigma' \to \Delta'$ be an arbitrary bijection. We extend f to a bijection $f' : \Sigma \cup \Sigma' \to \Delta \cup \Delta'$ for some Δ with $\Delta' \cap \Delta = \varnothing$ and $|\Delta| = |\Sigma - \Sigma'|$. Then

$$f(\Sigma_*'^* - L) = f'(\Sigma_*'^* - L) = \Delta_*'^* - f'(L) \in \text{co-CLS},$$

since f' is a bijective substitution from Σ to $f'(\Sigma)$. □

Not all picture classes are the complement of their complement. For instance, consider the set UNARY of unary languages, which is clearly a picture class. The language $L = \{1\}_*^*$ is in UNARY, so we have that $\{1\}_*^* - L = \varnothing$ is in co-UNARY. But then $\{0,1\}_*^* = \{0,1\}_*^* - \varnothing$ is in co-co-UNARY, so UNARY $\neq$ co-co-UNARY. Let us find a natural closure property of a class CLS that implies CLS = co-co-CLS.

Definition 3.38. Let CLS be a picture class. We say CLS is *literate* if for all $L \in$ CLS and Σ_*^* an alphabet, $L \cup \Sigma_*^* \in$ CLS and $L \cap \Sigma_*^* \in$ CLS.

Lemma 3.39. *Let CLS be a picture class. Then if CLS is literate, CLS = co-co-CLS.*

Proof. Clearly CLS $\subset$ co-co-CLS always holds. For the other direction, let CLS be literate, and consider $L' \in$ co-co-CLS. Then there exists $L \in$ CLS over some alphabet Σ, and two alphabets Σ_1 and Σ_2 such that $L' = (\Sigma_1)_*^* - ((\Sigma_2)_*^* - L)$.

We have

$$\begin{aligned}(\Sigma_1)_*^* - ((\Sigma_2)_*^* - L) = (\Sigma_1)_*^* - ((\Sigma_1 \cap \Sigma_2)_*^* - L) = \\ (\Sigma_1)_*^* - ((\Sigma_1 \cap \Sigma_2)_*^* - (L \cap (\Sigma_1 \cap \Sigma_2)_*^*)).\end{aligned}$$

Since CLS is literate, it is closed under intersections with full languages. We may thus find, given $L' \in$ co-co-CLS, even some $L \in$ CLS, Σ_1 and Σ_2 such that $\text{alph}(L) \subset \Sigma_2 \subset \Sigma_1$, and

$$L' = (\Sigma_1)_*^* - ((\Sigma_2)_*^* - L).$$

But then $L' = (\Sigma_1 - \Sigma_2)^*_* \cup L$, which proves the claim, since CLS is literate, and thus closed under unions with full languages. □

It is easy to check that DREC, REC and all the four-way automata classes are literate.

We now define proofs of exclusion for AFA, and prove they completely identify which pictures are in the languages of an automaton. Then, it will be easy to construct a recognizable grammar that draws such a proof of exclusion on the states. The grammar will, in fact, be even simpler than that in the NFA case.

Definition 3.40. Let $G = (V, E, r, \Sigma, \Gamma, l, f)$ be a labeled canvas. A *proof of exclusion* for an AFA A running on G is a possibly infinite rooted tree $T = (V_T, E_T, r_T, Q \times V, l_T)$ with vertices labeled over $Q \times V$ such that

- $l_T(r_T) \in I \times r$.
- no element of F occurs in a node label of T.
- all leaves carry a rejecting state.
- if $x \in V_T$ is an internal node with label $(q, v) \in (Q - U) \times V$, then for all $(q', g) \in \delta(q, l(v))$, x has a child with label (q', vg).
- if $x \in V_T$ is an internal node with label $(q, v_i) \in U \times V$, then for some $(q', g) \in \delta(q, l(v))$, x has a child with label (q', vg).

Lemma 3.41. *Let the AFA A have a unique initial state. Then A accepts $p \in \Sigma^*_*$ if and only if there does not exist a proof of exclusion for it.*

Proof. Let $p \in \Sigma^*_*$, and let r' be a proof of exclusion for it. We show p cannot be accepted by A. Suppose the contrary, and consider a hypothetical accepting run r of A. We will prove by induction on the size of subtrees rooted under the nodes of r, that none of the labels $l_r(v) = (q, x)$ for $v \in r$ can appear as labels of r'. This is easy to see for the leaves of r, for they have final states, and therefore cannot appear in a proof of exclusion.

Consider now a pair $l_r(v) = (q', y)$ for $v' \in r$ such that v' has at least one child v. Let $l_r(v) = (q, x)$. If q' is an existential state, then if (q', y) were to appear in r', also (q, x) would appear in r' by the local rules of r'. Therefore, $(q', y) \notin l_{r'}(r')$. If q' is universal, then if (q', y) were to appear in r', it would have at least one successor with respect to the transition function δ of A. This same successor (q'', z) would then be a successor of (q', y) in r by the local rules of accepting runs, and by induction, $(q'', z) \notin l_{r'}(r')$, implying $(q', y) \notin l_{r'}(r')$.

Since r will have an initial state in its root, we see that r' cannot have an initial state in the top left corner as the state of any pair that appears as a label. In particular, the root of r' cannot contain an initial state. This is a contradiction, which proves that proofs of exclusion in fact do prove noninclusion.

In order to prove the converse claim, let us assume p is not accepted by A. Let r' be the singleton tree with the label of the single node being (s, x) for s the initial state and x the top left corner.

We start building a tree with the local properties of proofs of exclusion as follows: Level by level, we build the tree r' in such a way that no partial

accepting runs can be rooted in its leaves. This is true initially, as we assumed $p \notin \mathcal{L}(A)$.

Consider a leaf (q, x) at some point of this process, and let q be an existential state. Then, if there exists a transition to an ID (q', y) such that a partial accepting run of A can have (q', y) in its root, then also (q, x) has this property. Similarly, if q is universal, if all transitions lead to an ID (q', y) such that there exists a partial accepting run rooted in (q', y), then also (q, x) has this property. Note that no leaf obtained can contain a final state, since otherwise that node, by itself, is an accepting partial run. Therefore, this process can be continued forever, although it is possible that all branches eventually end in a rejecting state, and the process stops adding levels.

Consider the limit point of the process, in the obvious sense. Clearly, this tree is a proof of exclusion, since we constructed it in such a way that all the local properties hold. □

Lemma 3.42. *If there exists a proof of exclusion r for automaton A and picture p, then there exists a periodic proof of exclusion for p as well, that is, a proof of exclusion r' with the additional property that for every two vertices $v, v' \in r'$ with the same label, the set of children of v and v' have the same labels.*

Proof. For each (q, x) occurring as a label in r, we choose the labels of the children of one such vertex v. We then deterministically start constructing a tree from the initial node. The limit of this process will clearly be a periodic proof of exclusion. □

Theorem 3.43. *co-AFA $\subset$ REC.*

Proof. Let A be an AFA with a unique initial state. We construct a grammar G such that every tiling contains a proof that there exists no accepting run of the given machine A on the picture p given. The grammar we construct only needs to store the current symbol and a subset of states of A in the states. Given p, the local rule puts an initial state at the top left corner of the picture of states $p' \in \Delta_*^*$. Accepting states of A are not allowed in any of the the states Δ. If the set of states of A stored in the state of cell x contains q, the interpretation is that there exists no accepting partial run of A rooted in (q, x).

As for the transitions, if the state set S in cell x contains an existential state q, then for all transitions $(q', g) \in \delta(q, l(x))$, the state q' must occur in the state set of xg. For universal states, for at least one $(q', g) \in \delta(q, l(x))$, the state q' is forced in the state set of xg. In both cases, the forced state–cell pairs are called the *successors*.

Let $p \in \Sigma_*^*$, and let p' be a locally consistent assignment of states. The top left corner of p' contains an initial state. It is clear by the local rules of the grammar that a proof of exclusion can be inductively built by following the successors of tiles. Therefore, $p \notin \mathcal{L}(A)$.

For the other direction, let $p \in \mathcal{L}(A)^c$. We will find a consistent assignment of states p' for its cells. Let r be a periodic proof of exclusion. For all (q, x) that occur as labels in r, we add the successor labels of such a vertex of r in the corresponding tiles. Since r contains the initial state in the root, and the local rules of the grammar are the same as those of proofs of exclusion, we find a consistent tiling, which proves $p \in \mathcal{L}(G)$. □

Since co-NFA is a subset of REC, but REC is not closed under complement, we must have the following.

Corollary 3.44. *NFA $\subsetneq$ REC.*

The following theorem is from [7], and can also be found in an earlier article by [12] with a rather long proof. We already proved the hard part in Section 2.3.

Theorem 3.45. *UFA $\bowtie$ co-AFA, UFA $\bowtie$ REC, AFA $\bowtie$ co-AFA and AFA $\bowtie$ REC.*

Proof. Note that since UFA $\subset$ AFA and co-AFA $\subset$ REC, it is enough to prove that UFA contains languages outside REC, and co-AFA contains languages outside AFA.

The language $L_{acyclic}$ of acyclic graphs is in UFA, because a UFA can check that, no matter how it moves along the edges of the graph, it will eventually reach a leaf. That is, we can construct a UFA that first universally moves onto every position of the picture, and then starts a search that universally quantifies over every outgoing edge, from every vertex. It only accepts when it reaches a leaf. It is clear that such a UFA has exactly $L_{acyclic}$ as its language.

By Theorem 2.63, $L_{acyclic}$ is not in REC. This means UFA $\not\subset$ REC.

On the other hand, suppose co-AFA $\subset$ AFA. Then by taking co-classes we get

$$\text{co-AFA} \subset \text{AFA} \implies \text{co-co-AFA} = \text{AFA} \subset \text{co-AFA},$$

so

$$\text{AFA} = \text{co-AFA} \subset \text{REC},$$

a contradiction.

This concludes the proof of the incomparability of the classes. □

Corollary 3.46. *AFA $\bowtie$ co-AFA.*

Proof. If AFA were a subset of co-AFA, then AFA would also be a subset of REC. If co-AFA were a subset of AFA, then by the previous proof AFA would again be a subset of REC. □

In particular, AFA is not closed under complementation.

We also obtain an interesting result for UFA, namely a concrete example of a language where UFA cannot be made always halting. In an intuitive sense, the following implies that UFA actually *need* to follow the cycles when recognizing $L_{acyclic}$.

Corollary 3.47. *There cannot exist an UFA for $L_{acyclic}$ that always halts.*

Proof. Suppose, on the contrary, that A is such a UFA. Then by Lemma 3.7, there exists an NFA A' for $L^c_{acyclic}$. Then, since the complement of every language in AFA is in REC by Theorem 3.43, we obtain that $(L^c_{acyclic})^c = L_{acyclic} \in$ REC, a contradiction. □

We are not aware of a 'nontrivial' automata theoretic characterization of REC in the sense of this chapter. However, the original definition of REC was in fact given under the name *online tessellation automata*, abbreviated OTA. These automata were defined by assigning states to positions, exactly like in REC, but instead of choosing all states at once and checking the local

constraints afterwards, the states were assigned in a predefined order, checking the local constraints (which were given by domino tiles) whenever possible. While this is a much more sensible way to implement a REC grammar as a computer program, the mathematical content is essentially the same.

Even though REC seems to lack a nice characterization by a picture walking automaton, the situation for DREC is much better. It turns out to be exactly the picture class 2WAFA^T. This class is easily seen to be the class of pictures recognized by AFA using a canvas representation where $\triangleright$ and $\triangledown$ are dropped. This means that when programming the automaton we can use universal and existential quantification, and an automaton can branch either up or to the left. The following theorem is from [11].

Theorem 3.48. *2WAFA^T = DREC.*

As usual, the proof is split into two inclusions. Given a picture p and a position y, we define *top left rectangle at y* to be the rectangle between $(1,1)$ and y. Given such a rectangle, we define its north child and west child as the rectangles that start from $(1,1)$ and end at $y-(1,0)$ and $y-(0,1)$, respectively.

Lemma 3.49. *$\text{2WAFA}^T \subseteq$ DREC.*

Proof. We make a DREC grammar G simulating the given 2WAFA^T A. At each state of G, the subset of states of A is held from which there is an accepting computation for the top left rectangle at the position. At the top left corner, the correct subset of states of A is enforced, based on the symbol at that position. We then inductively find the correct states elsewhere.

Using the intuition we gave for AFA acceptance, it should be clear that whether the AFA accepts a top left rectangle at y from a given state s only depends on the symbol at that position, and from which states it accepts the north and west children.

This can also be proved with the accepting run definition: Consider an accepting run from y and s. It is a tree with an initial segment of (s', y') tuples for some states s' and position $y' = y$ starting from the root. The leaves of this initial segment are either accepting leaves at y, or (s', y') tuples where y' starts either the north or west child of the rectangle at y. Therefore, whether such a tree exists is a function of which states the automaton accepts at y, and from which states the children of the rectangle are accepted.

Now, it is easy to make the grammar: At each position, for each state, we use the function given by the previous paragraph to determine whether there exists an accepting run from that state, and store the information in the tile. A state of G is accepting if there there exists an accepting run from at least one initial state of A. □

Lemma 3.50. *DREC $\subseteq$ 2WAFA^T.*

Proof. Given a DREC grammar G, we find a 2WAFA^T A with the same language. The automaton A has the states $\Delta \times S$ for the states Δ of G and a set of helper states S. Let us distinquish a state $i \in S$. When A is started in state (q, i) in cell x, it accepts if and only if there is a consistent assignment of DREC states onto the cells of the subpicture northwest from x such that q is used as the state cell x.

The top left corner can easily be detected by a 2WAFA^T. In the top left corner, A simply checks whether such a consistent assignment exists using a look-up table. In the general case, A guesses a preimage (q_1, q_2) of the given state q as the states of the north and west neighbors. It then recursively checks these that these subpictures can be consistently colored in such a way that q_i is used in the bottom right corner.

The algorithm works because in the recursion, in addition to consistent placements of states for the subpictures starting from the north and west neighbors existing, they must also have the same states in the overlapping zone due to the determinism of DREC.

Then, the initial states of A simply start such searches from all states $q \in \Delta$. □

3.4 NFA = FNFA

In this section, we will prove that the classes NFA and FNFA coincide. We do so by reducing the landing problem, that is, the problem of finding the possible landing cells and states of a partial run of a FNFA outside the picture, to a result of [7]. Namely, it turns out that an NFA accepting a picture language 'from the outside' can only recognize very simple languages, which are also in the class NFA, and that such outer NFA can be used to solve the landing problem.

As usual, instead of defining a new kind of nondeterministic finite state automaton, we instead define a new representation of a picture as a canvas, and run NFA with the existing definition on these representations.

Definition 3.51. Let ⊠ be a new special symbol not in any alphabet. Given picture $p \in \Sigma_*^*$, we define *outer representation* of p as $(V, E, r, \Sigma, \Gamma, l, f)$

$$V = \mathbb{Z}^2 - \mathrm{dom}(p)$$

$$l(v) = \begin{cases} \boxtimes & \text{if } v \in \mathrm{dom}(p) \\ \# & \text{otherwise.} \end{cases}$$

$$r = (0, 1)$$

The outgoing edges E and their road-coloring sets are the same as in the unbordered representation, except that if an edge would enter the domain of the picture, there is a self-loop with the same road-coloring set instead.

Definition 3.52. The picture class ONFA is the class of unary languages accepted by these nondeterministic automata, using the outer representation for the pictures being accepted.

It is clear that whether or not a picture is accepted depends on its shape only, which is why we only consider unary languages. Our definition of outer automata differs slightly from the one used in [7] in that in their model, the automaton can enter the (unary) picture, but must always exit on the next step, whereas our model senses the border, and cannot enter the picture. It is obvious that these models can simulate each other in a very local sense, and in particular that their languages must be the same. We choose to use the current definition, since in pictures of height 1, the top and bottom edges are the same.

It turns out we can find a rather simple characterization of ONFA. To do this, we represent picture shapes as words, and characterize the corresponding class of word languages.

Definition 3.53. The *shape word* of picture p is $0^{|p|}1\overline{p}$.

Before we state and prove our characterization of ONFA, we prove a *Landing Lemma* for partial runs that enter and leave a half-plane. We will need some auxiliary definitions.

Definition 3.54. The *empty configuration* is the unbounded representation of the empty picture.

Definition 3.55. For NFA A with a single initial state s, running on the empty configuration, we define its *south landing sequence* as $(s_i) : \mathbb{Z} \to 2^Q$ such that

$$\begin{aligned} s_i = \{s' \in Q \mid \exists \text{run } r \text{ of } A : \quad & r_1 = ((-1,0),s) \wedge \\ & r_{|r|} = ((0,i),s') \wedge \\ & \nexists\, 1 \le j < |r|, y, x, t : r_j = ((y,x),t) \wedge y = 0\}. \end{aligned}$$

The runs r in the definition of (s_i) are called *south landing runs*. Symmetrically, we define the north, west and east landing sequences and runs.

Lemma 3.56 (Landing Lemma)**.** *For any NFA A, a landing sequence is eventually periodic in both directions.*

Proof. We only show the result for the south landing sequence. Let s_i be the south landing sequence of A. It is enough to show that, for each state s', the binary sequence

$$s'_i = \begin{cases} 1 & \text{if } s' \in s_i \\ 0 & \text{if } s' \notin s_i \end{cases}$$

is eventually periodic in both directions. Then in each direction, (s_i) has the least common period of the periods of the sequences (s'_i), and its period starts when the periods of all (s'_i) have started.

To show this property for the sequence $(s'_i)_i$, let us first construct a PDA accepting the language L_{moves} of words $w \subset \{\triangle, \triangleright, \triangledown, \triangleleft\}^*$ such that some south landing run r of A has exactly this sequence as its sequence of moves, that is, $\text{seq}(r) = w$ for some south landing run r of A.

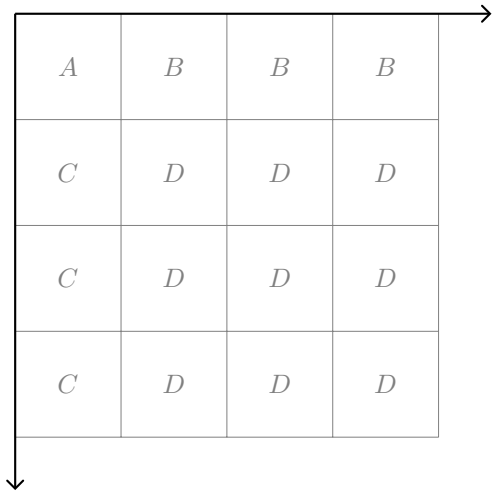
The PDA accepts when the stack becomes empty. It originally has one symbol Z on the stack, representing the fact that the NFA starts at height 1. The finite control makes its moves just as the NFA would, always reading the current move from the input word, rejecting the word if a different move is read. A Z' is pushed on top of the stack whenever the NFA moves up, and a Z' or Z is popped whenever it moves down. However, the automaton may only pop a Z if it is about to enter state s'. The stack is not changed when the NFA moves horizontally. It is clear that such a PDA accepts exactly L_{moves}, since when the stack become empty, the NFA being simulated would have entered state s', and would have, altogether, moved one step down from its initial position.

Because this language is context-free, its Parikh language L_p is a semilinear set. But then, by well-known closure properties of semilinear sets, also

$$\{y - x \mid (x, y, z, w) \in L_p\}$$

is semilinear, where x and y are the amounts of left and right moves, respectively. But this is exactly the set $\{i \mid s'_i = 1\}$, and a semilinear subset of $\mathbb{Z}$ must be eventually periodic in both directions. □

Figure 8: An eventually periodic subset of $\mathbb{N}^2$.



Definition 3.57. For an NFA A, we define the fundamental landing period from state s as the least common period of the periods of landing sequences of A when s is used as the unique initial state of A. We define the fundamental landing threshold t from state s as the smallest number such that all periods of landing sequences start at a position with absolute value less than or equal to t. Now, the *fundamental landing period* and *fundamental landing threshold* are obtained by doing this for every state s', taking the least common period of the periods, and the largest of the thresholds. For brevity, we will usually omit the word 'landing' in these terms.

Definition 3.58. A subset X of $\mathbb{N}^2$ is eventually periodic if there exist t and q such that

$$\forall w > t : (h, w) \in X \iff (h, w + q) \in X$$

and

$$\forall h > t : (h, w) \in X \iff (h + q, w) \in X.$$

We may always take t and q to be equal by for instance choosing both to be qt, since $q|qt$ and $qt \geq t$. Figure 8 illustrates an eventually periodic subset of $\mathbb{N}^2$. The quarterplane is partitioned into square blocks of the same size, and the four square blocks at the top left corner are repeated as in Figure 8 to determine the contents of the whole quarterplane. We call a tuple (A, B, C, D) containing such blocks the *block representation* or an eventually periodic set.

Note that the two dimensional semilinear sets are a proper superclass of the eventually periodic sets.

Lemma 3.59. *Let $X \subset \mathbb{N}^2$, and let t and q be such that*

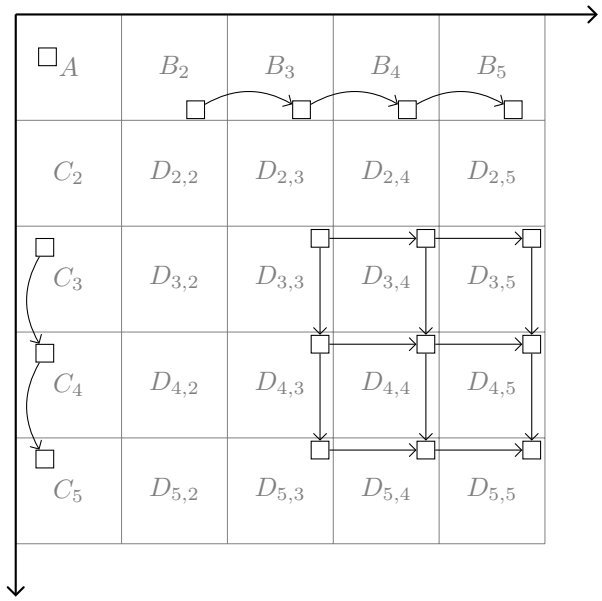
$$w > t \wedge (h, w) \in X \implies (h, w + q) \in X$$

and

$$h > t \wedge (h, w) \in X \implies (h + q, w) \in X.$$

Then X is eventually periodic.

Figure 9: A filled cell forces cells to be filled in all squares of the same type southeast of the current square.



Proof. We may assume $t = q$ by substituting qt for both t and q. This gives us a nice picture of the situation, depicted in Figure 9. We partition the quarterplane into square blocks of the same size, and further partition these blocks into A-blocks, B-blocks, C-blocks and D-blocks, as in Figure 9. We have the filling rule that once a cell is filled in one of these blocks, the block tells its east and south neighbors to fill the same cell, if they are the same type of blocks. The situation differs from that of Figure 8 in that the blocks of the same type need not be the same. Instead, it is only required that the blocks are 'nondecreasing'.

First, we note that each row and column must be eventually periodic: Consider row h. The claim is clear if this row is empty. Otherwise, since there are only a finite amount of numbers (mod q), we may take

$$m = \max\{\min\{n \mid n > t, n \equiv j \pmod q, (h,n) \in X\} \mid 0 \leq j < q\}$$

where the interpretation of $\min\{\}$ inside a set construction is that no new element is added.

If $w > m$ ($\geq t$), and $(h,w) \in X$, then also $(h, w - kq) \in X$ for some k such that $w - kq > t$, since $m < w$: otherwise w would have been the smallest number greater than t in its modulo class, in which case we would necessarily have $m \geq w$. Therefore also $(h, w-q) \in X$ due to the assumption $(h,w) \in X \wedge w > t \implies (h, w+q) \in X$. The claim for columns is symmetric.

Next, let us show that if both h and w are large enough, and $(h,w) \in X$, then both $(h, w-q)$ and $(h-q, w)$ are in X. We will then show the statement of the theorem follows from these two results.

Consider a binary tile M obtained from one of the D-blocks b by coloring cells of X with 1 and others with 0 such that the number of 1's is maximal among

all such tiles. Let c be the index of the column to the right of b, let r be the index of the row under b, and let $d = \max\{c, r\}$. Now, consider $(h, w) \in X$ such that $h > d \wedge w > d$. Since (h, w) is in a D-block e to the southeast of b, the tile N corresponding to e must be pointwise greater than or equal to M, and thus equal to it by the maximality of M. Therefore, necessarily $(h-q, w) \in X \wedge (h, w-q) \in X$.

Now, for the d defined in the previous paragraph, take $t' = d + q$. For each r of the finitely many rows not intersecting with $[t', \infty) \times [t', \infty)$, we take the least common threshold t_r and the least common period q_r for the eventually periodic sequences on those rows. Similarly, we obtain t_c and q_c for the columns. Now, clearly if $(h, w + q) \in X \wedge w > \max\{t', t_r\}$, then also $(h, w) \in X$, since if $h \leq t'$, $w > t_r$ implies the row has already become periodic (since all rows with $h < t'$ have), and therefore $(h, w+q) \in X \implies (h, w) \in X$. If, on the other hand, $h > t'$, then both h and $w + q$ exceed $t' = d + q$, and therefore $(h, w) \in X$. The claim for columns is symmetric, with threshold $\max\{t', t_c\}$ for h.

Therefore, $\max\{t', t_r, t_c\}$ is the threshold required, and q can be taken as the period. This implies X is eventually periodic. □

Before proving the next characterization of the eventually periodic sets, let us state the following version of the pumping lemma for regular languages. We skip the proof, but one can be found in [20].

Lemma 3.60. *Let L be a regular language. Then there exists t such that if*

$$uvw \in L \wedge |v| > t$$

then

$$\exists x, y, z : v = xyz \wedge |y| \geq 1 \wedge (\forall k \geq 0 : uxy^k zw \in L)$$

Lemma 3.61. *A subset X of $\mathbb{N}^2$ is eventually periodic if and only if the language*

$$\{0^n 1^m \mid (n, m) \in X\}$$

is regular.

Proof. First, consider a regular language

$$L \subset \{0^n 1^m \mid (n, m) \in X\},$$

and a word $w = 0^i 1^j \in L$. Let t be the threshold of L. Then, if $i > t$, we have $0^{i+n*t!/n} 1^j = 0^{i+t!} 1^j \in L$ for some $n > 0$ by the pumping lemma, and similarly for 1's. This means the corresponding subset of $\mathbb{N}^2$ has the pumping property of Lemma 3.59 with period $t!$ and threshold t.

Now, consider an eventually periodic subset S of $\mathbb{N}^2$. We may again assume the threshold t and period q of S are equal, and thus the situation is that of Figure 8. The automaton we construct can be thought of as a finite automaton that is first moved some distance to the right, then some distance down, and is then asked if the current position is in S.

But clearly such an the automaton can keep track of the type (A, B, C or D) of block it is in, and its position in such a block, that is, its position modulo q on each axis. Therefore, the automaton can easily determine if the shape of p is in S. □

Theorem 3.62. *ONFA is exactly the class of unary picture languages L such that shape(L) is eventually periodic, that is, the language of shape words of pictures in L is regular.*

Proof. To prove this theorem, we consider a language L' in ONFA, and let $L = \{(\overline{p}, |p|) \mid p \in L'\}$, which uniquely determines the language. We prove both the widths and heights of pictures in the language can be pumped, that is, there is a threshold t and a period q such that

$$\forall (h, w) \in L : h > t \implies (h + q, w) \in L$$

$$\forall (h, w) \in L : w > t \implies (h, w + q) \in L$$

It is enough to find such t and q that work for widths, since a symmetric argument will work for heights, and we can use the larger of the thresholds and a least common period of the periods to find the threshold and the period of the whole language. Then, the claim follows from Lemma 3.59 and Lemma 3.61.

Let q' and t' be the fundamental period and fundamental threshold of A. We will prove that the period and threshold of L are $q = (|Q|t')!q'$ and $|Q|t'$, respectively. We start by naming some areas of the plane. We define the top left ray, the top right ray, the bottom left ray and the bottom right ray as the sets of cells

$$\{(0, -n) \mid n \in \mathbb{N}_0\},$$

$$\{(0, |p| + n + 1) \mid n \in \mathbb{N}_0\},$$

$$\{(\overline{p} + 1, -n) \mid n \in \mathbb{N}_0\},$$

and

$$\{(\overline{p} + 1, |p| + n + 1) \mid n \in \mathbb{N}_0\},$$

respectively. See Figure 10. Note that the top and bottom rays do not coincide even if the picture has height 1.

Now consider a run r accepting the picture of shape (h, w), where $w > t$. We split the run into partial runs $r^1, \ldots, r^k$ between the points of intersection with one of the four rays. That is, let I be the sequence such that $I_1 = 0$ and $(I_2, \ldots, I_{|I|-1})$ are the indices i of the ordered subsequence of r such that r_i is on top of one of the rays, and $I_{|I|} = |r|$. Then

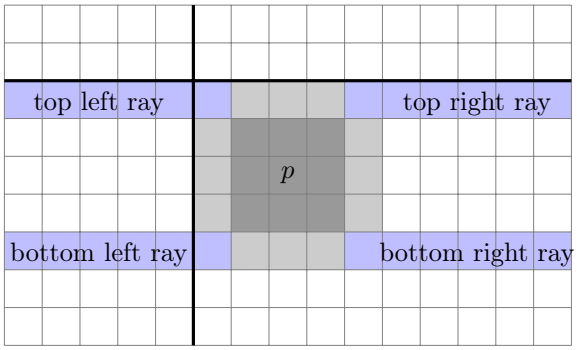
$$r^i = (r_{I_i+1}, \ldots, r_{I_{i+1}}).$$

In other words, we partition the run into semiopen intervals open to the left by cutting the run at the points where a ray is crossed. The last interval of the r^i holds the remaining part of r, and it is empty if the automaton accepts on a ray.

These partial runs are transformed into a run of the automaton on the picture of shape $(h, w + q)$. There is an obvious way to identify the rays of the small picture with the rays of the large picture, and each partial run is transformed into a run that starts and ends at the corresponding cell in the corresponding ray. It is then clear that by gluing together the partial runs obtained for the larger picture, a valid accepting run for the larger picture is obtained, proving $(h, w + q) \in L$.

For partial runs between two cells both in left rays, or two cells both in right rays, it is obvious how to find a corresponding run for the larger picture: Partial

Figure 10: Named areas outside the picture.



runs on the left side are valid as such, and partial runs on the right side are translated q cells to the right. Then, the starting points and the ending points of the transformed partial runs are at the cells of the rays of the large picture corresponding to the starting and ending cells of the original partial runs on the rays of the small picture. These statements are based on the fact that the small and large picture cannot be distinguished by a partial run that doesn't touch a ray on both sides of the picture (because the subgraph seen will be identical on both pictures).

Now, all we need to do is apply the Landing Lemma 3.56 to transform the partial runs that move between the two sides of the small picture to such runs on the large picture. For this, consider such a partial run $r^j = u$ on the small picture. By symmetry, we may assume u starts just above a cell of the top left ray, and ends on a cell of the top right ray. We split u further into subpartial runs u^j exactly as we did for r, but this time at the positions where the automaton reads a $\boxtimes$. Let J the sequence of indices of u where the automaton reads a $\boxtimes$.

We now have two cases to consider:

- One of the subpartial runs u^j moves the automaton to the right more than t' cells.
- All subpartial runs u_j move the automaton at most t' to the right.

In the first case, we are done, since the automaton's fundamental threshold has been exceeded, so the subpartial run u^j can be made mq' cells longer to the right, for any $m \in \mathbb{N}$. In particular, it can be made $q = (|Q|t')!q'$ cells longer. That is, the partial run u can be converted from one on the picture of size (h, w) to one on the picture of size $(h, w + q)$ by changing one of its subpartial runs to a longer one.

In the latter case, we note that there must be more than w/t' of these subpartial runs, where we chose $w > t = |Q|t'$. Then, we may take an ascending subsequence K of J such that

- the sequence of positions of $(u_i)_{i \in K}$ is ascending to the right.
- between each two indices of K the automaton has moved at most t' cells to the right.

- $|K| > |Q|$.

Now note that there must be a repetition of states in $(u_i)_{i \in K}$, say at $a, b \in K, a < b$, and the distance d between u_a and u_b satisfies $1 \leq d \leq |Q|t'$. Then the partial run of length d between u_a and u_b can be repeated $\frac{q}{d} = \frac{(|Q|t')!q'}{d}$ times to obtain a partial run for the picture $(h, w + q)$.

This means that in both cases, we were able to make the partial run work for a picture q wider, as long as the picture had width at least t, and thus t and q as we chose them are a threshold and period for pumping the widths of picture in L. □

Corollary 3.63. *ONFA ⊂ NFA.*

Proof. Let an ONFA A with language L be given. Then, the shapes of pictures in L form a semilinear set, and therefore there is even a 2WNFA A' with the same language by Theorem 3.24, since all eventually periodic sets are semilinear. □

In order to justify certain symmetry arguments in the future, we note that it is clear that ONFA could just as well have been defined with the same graph representation but with some other corner of the picture as the root of the canvas. This is because an ONFA can deterministically change the corner in which it starts its run, and therefore ONFA starting from one corner can simulate ONFA starting from another corner.

Given a picture, we call the cells (i, j) such that $j < \text{left}(p)$ the west half-plane H_w, and similarly we get the east, north and south half-planes H_e, H_n, H_s. These cover all the positions outside the picture. Given a run of an FNFA starting from just outside the edge of a picture and ending at an edge, staying outside the picture during the run, we call the unique half-plane on which it starts the *initial half-plane*. The part of the run before exiting the initial half-plane for the first time is called the *initial segment* of the run. Similarly, we define the *final half-plane* and the *final segment* of the run.

Again, we need to name some parts of a given picture p and the plane outside it. For the purposes of the following arguments, it is more convenient to redefine rays to start from the corners of p instead.

Definition 3.64. Given picture p, we define the rays as

$$R_{tln}(p) = [0, -\infty) \times \{1\},$$

$$R_{tlw}(p) = \{1\} \times [0, -\infty),$$

$$R_{trn}(p) = [0, -\infty) \times \{\text{right}(p)\},$$

$$R_{tre}(p) = \{1\} \times [\text{right}(p) + 1, \infty),$$

$$R_{bls}(p) = [\text{bottom}(p) + 1, \infty) \times \{1\},$$

$$R_{blw}(p) = \{\text{bottom}(p)\} \times [0, -\infty),$$

$$R_{brs}(p) = [\text{bottom}(p) + 1, \infty) \times \{\text{right}(p)\}$$

and

$$R_{bre}(p) = \{\text{bottom}(p)\} \times [\text{right}(p) + 1, \infty).$$

In R_{xyz}, x determines whether the ray starts at a top or a bottom corner, y determines whether the corner is a left or a right corner, and z determines the

cardinal direction to which the ray extends. We talk about, for instance, the top left north ray, to indicate R_{tln}. Syntactically, we only need R_{tln} and R_{bls} in the proofs due to heavily relying on the 8-fold geometrical symmetry of the set of rectangles.

Definition 3.65. Given $p \in \Sigma_*^*$, the left edge $E_l(p)$ of p is the set

$$\{(1, \text{left}(p)), \ldots (\underline{p}, \text{left}(p))\}.$$

Similarly, we define the top, right and bottom edges E_t, E_r and E_b, respectively. The set of edge cells of p is

$$E(p) = E_t \cup E_r \cup E_s \cup E_w.$$

$C(p, m)$ is the set of edge cells of p at most distance m away from the corners of p.

Definition 3.66. The left line of p is the union of the top left north ray, the bottom left south ray and the left edge of the picture. Similarly, we get the top line, the right line and the bottom line of p.

Definition 3.67. The set of cells outside the set $X \subset \mathbb{N}^2$ is

$$O(X) = \mathbb{Z}^2 - X.$$

For picture p, we define $O(p) = O(\text{dom}(p))$.

We are interested in characterizing the states and cells in which an NFA can enter a region. Let us formalize this idea.

Definition 3.68. Let Q be a finite set of states. Then, a *state configuration on Z over Q* is a sequence $(2^Q)^Z$ for some set $Z \subset \mathbb{Z}^2$. The set Q is often clear from the context, and is left unspecified.

Definition 3.69. Let A be an NFA with states Q. Then, if $X, Y, Z \subset \mathbb{Z}^2 \times Q$ we define

$$A(X, Y, Z) = \{z \in Z \mid \exists \text{ run } r \text{ of } A : r \in XY^*z\}$$

referred to as a set of *landings*. We also use the syntactic conventions that if a tuple is used in place of one of X, Y, Z, the tuple is enclosed in a singleton set, which is then used as the argument. If one of X, Y, Z is not a subset of $\mathbb{Z}^2 \times Q$, but a subset of $\mathbb{Z}^2$, then its cartesian product with Q is used instead.

Of course, if W is a subset of $\mathbb{Z}^2 \times Q$ such that $\pi_1(W) \subset Z$, then we naturally obtain a state configuration c on Z from W by defining

$$\forall x \in Z : c(x) = \{q \mid (x, q) \in W\}.$$

The following two are useful special cases of the general definition of landings.

Definition 3.70. Let Q be a finite set of states. For any $Z \subset \mathbb{N}^2$ and FNFA A with unique initial state s, the *landing sequence of A from x onto Z* is defined as the state configuration c on Z corresponding to

$$A((x, s), O(Z) \times Q, Z \times Q).$$

Let A be an FNFA with state set $Q' \supset Q$ and with a unique initial state $s \in Q' - Q$. We say *A computes the state configuration c from x* if c is the state configuration on Z corresponding to

$$A((x,s), Z \times (Q' - Q), Z \times Q).$$

Lemma 3.71. *Let A be an FNFA with states Q. Then, for every $i \in Q$ and $m > 0$ there exists an NFA A' with states $Q' \supset Q$ such that*

$$\forall p : A(((0,1), i), O(p), C(p,m)) = A'(((1,1), s'), dom(p), C(p,m) \times Q)$$

Proof. $C(p,m)$ is a finite set, so A' simply needs to guess one of the landing possibilities $(x,s) \in C(p,m) \times Q$ and check if it's a possible landing of A. If it is, A' finds x and enters s on it. But it's easy to check if (x,s) is a landing of A: We construct an ONFA B that simulates A until it tries to enter the picture, and accepts if it would've entered the correct cell x in the correct state s. The set shape$(\mathcal{L}(B))$ is eventually periodic, so A' can check if p belongs to $\mathcal{L}(B)$. Since $p \in \mathcal{L}(B)$ if and only if (x,s) is a landing of A, we are done. □

Lemma 3.72. *Let A be an FNFA with states Q. Then, exists NFA A' with states $Q' \supset Q$ such that*

$$A(((0,1), i), O(p), E(p)) = A'(((1,1), s'), dom(p), E(p) \times Q)$$

Proof. Let t, q be the fundamental threshold and period of A, respectively, and assume $(x,s) \in Z = A(((0,1), i), O(p), E(p))$. First assume $x \in E_l(p)$. Then

$$x \notin C(p, t+q+1) \implies (x + (q,0), s) \in Z$$

by the Landing Lemma 3.56, since A's run with landing (x,s) must have entered the west half-plane for the last time at some point, and we may thus pump the final segment of the run before landing, by q steps in either direction. By repeating the argument, (y,s) is also in Z for some $y \in C(p,t+q+1) - C(p,t+1)$.

Conversely, if (y,s) is a landing of A onto $C(p,t+q+1) - C(p,t+1)$, on the west side, then also all other $(y+(mq,0), s)$ such that $y+(mq,0) \in E(p) - C(p,t+1)$ are landings of A, again by the Landing Lemma 3.56, in particular if y is obtained from pumping x onto $C(p,t+q+1) - C(p,t+1)$, then the landing (x,s) is found by pumping y the same distance in the other direction.

Now, for landings of A on $C(p,t+q+1)$, we apply Lemma 3.71. As for other landings, consider again landings on the west side of p. For all landings of A onto (x,s), where $x \in C(p,t+q+1) - C(p,t+1)$, we let A' move q steps up or down as many times as it likes before entering s, while staying inside $E(p) - C(p,t+1)$. By the argument of the first paragraph, we obtain all landings of A on E_l this way. The other edges are similar. □

Theorem 3.73. *NFA = FNFA.*

Proof. Given FNFA A, we construct an NFA A' that has states $Q \cup Q'$ where Q are the states of A and Q' are helper states used in the simulation. The NFA A' uses only states of Q when inside p, and has the same transition function as A when restricted to these states. Of course, this means A might try to exit p during its accepting run of p. When it tries to exit p, x being the first cell

outside p it would've entered, it instead enters a special search state in Q' that computes the landing sequence of A' from x onto the edges of the picture inside $\mathrm{dom}(p)$. We may assume A never accepts a picture p outside $\mathrm{dom}(p)$, since an NFA can use existential to find the picture, and then accept inside its domain.

If we can accomplish this, then it should be clear that the languages of A and A' are the same. This is because accepting runs of A can be turned into accepting runs of A' on the same picture by turning all partial runs outside the picture into ones using the states Q'. On the other hand, if r is any run of A', we can take the subsequences using states of Q', and turn them into runs outside the picture.

We do not give a formal construction of A', but an informal algorithm for obtaining such an automaton from the behavior of A. So assume A exits p during a run, and let x be the first cell outside p it sees. We may assume x is on the west side of p, since the other cases are symmetric. First, we note that

$$\begin{gathered}A((x,s),O(p),E(p)) = A((x,s),H_w,E_l)\cup \\ A(A((x,s),H_w,R_{tln}),O(p),E(p))\cup \\ A(A((x,s),H_w,R_{bls}),O(p),E(p))\end{gathered}$$

We explain how A' can compute each of the three parts, in which case the pointwise union is also computable. By symmetry, it's enough to handle $A((x,s),H_w,E_l)$ and $A(A((x,s),H_w,R_{tln}),O(p),E(p))$.

Case 1: Computing $A((x,s),H_w,E_l)$ inside $\mathrm{dom}(p)$.

Consider the landing sequence s^1 of A from (x,s) upwards, that is, s^1_i is the set of landing states on the cell $x+(-i,1)$. By the Landing Lemma 3.56, this sequence s^1 is eventually periodic. Therefore A' can compute this sequence inside E_l, since the sequence does not depend on the picture. This concludes the first case.

Case 2: Computing $A(A((x,s),H_w,R_{tln}),O(p),E(p))$ inside $\mathrm{dom}(p)$.

Let $x=(i,j)$. Then,

$$A((x,s),H_w,R_{tln}) = s^1_{[i+1,\ldots)} = s^2,$$

where s^1 is as in Case 1. Let B_{s^2} be an FNFA with initial state s'' that computes this sequence onto R_{tln}, and then simulates A. That is, B_{s^2} has states $Q\cup Q''$, with Q'' and Q disjoint,

$$B_{s^2}(((0,1),s''),R_{tln}\times Q'',R_{tln}\times Q) = A((x,s),H_w,R_{tln})$$

for the unique initial state s'' of B_{s^2}, and B_{s^2} has the same transition function as A, when restricted to states in Q, which implies

$$B_{s^2}(((0,1),s''),O(p),E(p)) = A(A((x,s),H_w,R_{tln}),O(p),E(p)).$$

By Lemma 3.72, there is an inner NFA with the same landings as B_{s^2}. Therefore, A' can move up the side of p, determine the sequence s^2, and depending on this sequence, compute the landing sequence of B_{s^2} onto $E(p)$ inside $\mathrm{dom}(p)$. This is possible, because the sequence s^2 is a final segment of the eventually periodic sequence s^1, and thus one of a finite amount of possibilities. This concludes the second case.

□

4 Picture expressions and generating devices

In Chapter 3 we saw that finite state automata that read cells one by one, changing their position according to a local rule, do not characterize the recognizable languages, even though both of the corresponding ideas give exactly the regular languages in the one dimensional case. The class of 'regular picture languages' is a third attempt at lifting the idea of one dimensional regular languages to two dimensions. Not surprisingly, this class, while having a natural definition, does not correspond to any of the classes defined sofar, as far as we know.

In this chapter, we define the class RE of regular picture languages and some variants of it. In particular, this class is shown to be mutually incomparable with REC. We also define another picture class RLG using a generating device, and compare it with some of the classes defined by tilings and automata.

The new results of this chapter, as far as we know, are the connection between RLG and DREC and the characterization of RLG by certain restricted finite state automata. We also give a new proof for the mutual incomparability of RE and REC, although arguable ours is much less sophisticated than that of Oliver Matz [13].

4.1 Definitions and connections between classes

Picture expressions are essentially syntax for giving languages in picture classes defined by closure properties. Similarly, in the case of words, regular expressions are syntax for the languages in the smallest class of word languages that the finite languages and is closed under concatenation, Kleene star and union. For instance, we could just as well have defined picture expressions for FOC in Section 2.4.

Having an expression syntax gives a nice and concise way to build languages, and often even REC languages are given using a regular expression. In fact, we have already introduced all of the operations used in the regular expressions, and given regular expressions as proof of an REC language existing. Another reason we formalise the notion of picture expressions instead of defining these classes by closure properties is to justify using induction on expression size. In most cases, either closure properties or induction can be used to prove the same things about a picture class. Often, it is easier to work with closure properties when proving a property is preserved by an operation, but when an exact numerical result is needed, induction makes the argument clearer.

In the following definition, the expressions are just syntax. For instance, $r_1 \ominus r_2$ is not evaluated in any way, but is just a string of symbols instead (the usual parsing rules apply though).

Definition 4.1. Regular expressions over Σ can be defined inductively:

- For each $a \in \Sigma$, a is a regular expression, and the empty set $\varnothing$ is a regular expression.
- If r is a regular expression, then r^c, $r^{*\ominus}$ and $r^{*\oplus}$ are regular expressions.
- If r_1 and r_2 are regular expressions, then $r_1 \ominus r_2$, $r_1 \oplus r_2$, $r_1 \cap r_2$ and $r_1 \cup r_2$ are regular expressions.

That is, regular expressions are exactly the well-formed expressions built from atoms, and the operators $\{^c, ^{*\ominus}, ^{*\oslash}, \ominus, \oslash, \cap, \cup\}$. Also the language $\mathcal{L}(r)$ of a regular expression r is defined recursively, in the obvious way. The class of languages obtained this way is denoted by RE.

The full language with respect to which complements are taken has to be inferred from context. Usually, it is obvious what is meant. Since we want the operations to be closure properties, one is, in theory, allowed to use complements with respect to multiple alphabets within the same expression. Of course, it is easy to see that expressions limited to one alphabet for the complement operation are no less powerful due to the fact RE is closed under the intersection operation and contains the full languages.

The previous definition is from [1]. It is worth noting that the terminology is not standard when it comes to regular expressions for picture languages. The following definition is advocated by Oliver Matz as the 'correct' definition of regular expressions for picture languages. We give it another name to avoid confusion.

Definition 4.2. The simple regular expressions, SRE, is the family of languages given by regular expressions using the operations $\{^{*\ominus}, ^{*\oslash}, \ominus, \oslash, \cup\}$.

Definition 4.3. CFRE is the family of languages given by regular expressions without complement.

Theorem 4.4. *SRE* $\subset$ *CFRE* $\subset$ *REC.*

Proof. REC contains the singleton languages, and has all the operations used in CFRE expressions as closure properties. □

Definition 4.5. SFRE is the family of languages of regular expressions that don't use the operators $\{^{*\ominus}, ^{*\oslash}\}$.

In the case of SFRE, it is even more crucial to know the alphabets with respect to which complements are taken. Luckily, it turns out even if the whole expression needs to take its complements with respect to the same alphabet Σ, we can always construct an expression that takes the complement with respect to a subset of Σ. This easily follows from the following.

Lemma 4.6. *Let* $A \subset \Sigma$ *and let all complements be taken with respect to* Σ. *Then* A^*_*, *is in SFRE.*

Proof. The language Σ^*_* is in SFRE by the expression $U = \varnothing^c$. To remove all pictures containing a given symbol a from the language of the regular expression E, the expression

$$E \cap (U \ominus (U \oslash a \oslash U) \ominus U)^c$$

can be used. □

Later, we will also need the fact SFRE contains the full column and row languages $\Sigma^{*\oslash}$ and $\Sigma^{*\ominus}$.

Lemma 4.7. $\Sigma^{*\oslash} \in$ *SFRE and* $\Sigma^{*\ominus} \in$ *SFRE.*

Proof. Let U be an SFRE expression for Σ_*^*. For all pairs $(a,b) \in \Sigma^2$, construct $E_{ab} = U \ominus (U \oplus ab \oplus U) \ominus U$. Then $(\bigcup_{(a,b)\in\Sigma^2} E_{ab})^c$ is a star-free regular expression for the language of pictures over Σ with exactly one column. The case of rows is symmetric. □

The class RLG is named after the right-linear grammars in the theory of one-dimensional languages. We give it a rather different definition than that of [1]'s, and ours does not refer to grammars at all. It is easy to see, though, that the two definitions give the same picture class.

Definition 4.8. A right-linear grammar G is a tuple $(L, (L_d)_{d\in\Delta})$, where L is a one-dimensional regular language over an intermediate alphabet Δ, and for each $d \in \Delta$, $L_d \subset \Sigma^*$ is also regular, for some alphabet Σ common to all the L_d. A picture p in the language $\mathcal{L}(G)$ of G is generated by choosing some $w \in L$ and $n \in \mathbb{N}$, and for each $1 \le i \le |w|$ taking a word of length n in L_{w_i} as p's ith column. The family of languages obtained this way is called RLG.

As is often the case, when simulating RLG by another device, it is nice to restrict to RLG's of a restricted (but universal) form. Such a natural normalized form is given in the following lemma.

Lemma 4.9. *For each RLG grammar $G = (L, (L_d)_{d\in\Delta})$, there is an RLG grammar $G' = (L', (L'_d)_{d\in\Delta'})$ such that $\mathcal{L}(G) = \mathcal{L}(G')$, and all the languages L'_d are disjoint.*

Proof. As the intermediate alphabet, we take $\Delta' = 2^\Delta$, and for every $D \in \Delta'$, take the regular language

$$L_D = \bigcap_{d\in D} L_d \cap \bigcap_{d\notin D} L_d^c.$$

Then it is clear that L_D are disjoint, and that if we take a regular expression r for L, and for all $d \in \Delta$ change all occurrences of d in r to $(\sum_{d\in D} D)$, we obtain a regular expression over Δ' with language L' such that the RLG grammar $G' = (L', (L'_D)_{D\in\Delta'})$ has the same language as G. □

In [1], it is proven that DFA is a proper superset of RLG. We prove a characterization of RLG by certain restricted DFA instead, and obtain the (proper) inclusion of RLG in DFA as a corollary. As usual, we use the approach of defining a new way of representing a picture as a canvas.

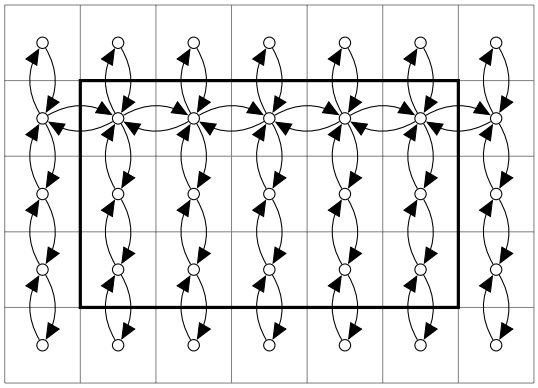
Definition 4.10. Given $p \in \Sigma_*^*$, the *comb representation* or p is the same as its bounded representation, except that outgoing edges with labels $\triangleleft$ and $\triangleright$ are changed into self-loops outside positions (i,j) such that $i \neq 1$. See Figure 11 for an illustration of this.

Definition 4.11. For all the automata types XFA defined, CXFA is the picture class of languages L such that there is an XFA A accepting the comb representations of exactly the pictures in L.

It is obvious that CXFA $\subset$ XFA for all the automata classes defined.

Note that the comb representation of a picture can be thought of as an undirected binary tree. The languages of DFA and NFA in the sense we define them are known to be different on trees in general, see [14]. However, since our trees are of a very restricted form, we can prove the following characterization of RLG, which also shows CDFA and CNFA coincide.

Figure 11: The comb representation of a picture. Self-loops are not drawn, and the solid line represents the border of the picture.



Theorem 4.12. *CDFA = RLG = CNFA.*

Proof. We reduce the problem to the result that both DFA and two-way NFA recognize exactly the regular languages, and in fact deduce the claim by characterizing both RLG and CXFA by certain projections of their picture languages, and noting that the corresponding one dimensional languages are the same.

First, let us define a two-way correspondence $f : \Sigma_{**}^{*} \to L_{pics}$ between pictures over Σ and words in the language

$$L_{pics} = \bigcup_{n,m \in \mathbb{N}} (\Sigma^n|)^m.$$

It is clear that we get a word of L_{pics} for each picture by listing the columns with $|$ appended. Conversely, the picture p corresponding to $w_1|\cdots w_m| \in L_{pics}$, where $w_i \in \Sigma^*$, is defined by $p[i, *] = w_i$. These operations are clearly inverses of each other, showing f is a bijection.

Next, we show that if a picture language L is in RLG by a grammar $(L_{int}, (L_d)_{d \in \Delta})$, then $f(L)$ is the intersection of L_{pics} and a regular language L'. To show this, take the nondeterministic substitution h such that

$$\forall d \in \Delta : h(d) \in (L_d|).$$

The language $h(L_{int})$ is regular, because the regular languages are closed under nondeterministic substitutions, and the intersection $h(L_{int}) \cap L_{pics}$ is of course exactly $f(L)$.

Conversely, let $L \subset (\Sigma^*|)^*$ be a regular language accepted by a one-way DFA A with states Q and final states F, and let $L' = L \cap L_{pics}$. We want to show that $f^{-1}(L')$ is in RLG. For each function $g : (Q - F) \to Q$, consider the language $L_g \subset \Sigma^*$ containing the words $w \in \Sigma^*$ such that for all s, if A is started from state s, then it accepts while reading $w|$ if and only if $g(s) \in F$, and otherwise it exits $w|$ in state $g(s)$. Note that L_g is clearly regular. We take the set of such functions as the set Δ, and for each $g \in \Delta$, use the language L_g as the language corresponding to g.

Now consider the intermediate language L_{int}. If we can find a language $L_{int} \in \Delta^*$ such that $h(L_{int}) = L$, for the same substitution h as before, then $G = (L_{int}, (L_d)_{d \in \Delta})$ defines the correct RLG, because

$$f(\mathcal{L}(G)) = h(L_{int}) \cap L_{pics}$$

by how the language of an RLG is defined, and

$$h(L_{int}) \cap L_{pics} = L \cap L_{pics}$$

as required.

Such a language L_{int} is easy to construct, we will give a DFA A' accepting this language. The DFA has the same set of states as A, but different transitions. Namely, if it reads g in state s, it changes its state to $g(s)$ if $g(s) \notin F$ and otherwise accepts the word. Then it is clear that A''s accepting runs on words over Δ correspond to accepting runs of A on words of $h(\Delta^*)$ due to how Δ and h were defined.

Next, we characterize the pictures accepted by CDFA and CNFA in the exact same fashion. We first show that if L is in CNFA by automaton A, then $f(L) = L' \cap L_{pics}$ for some regular language L'. As usual, we assume A always directly returns from a border cell to $\mathrm{dom}(p)$. We make a two-way NFA A' accepting such a language. First, note that if $f(p) = w$, cells of p naturally map onto cells on w, leaving only the |-cells of w without a 'preimage'.

The automaton A' we construct has the states $Q \times S$ for a set of search states S, where Q is the state set of A. When the S-side of the cartesian product is s_{sim}, A' simulates A as if it the word w it is accepting was the projection of a picture p, and as if it were on the cell of w corresponding to the cell of p A is on (as long as A stays inside p).

This means

- If A moves up and it is not at the top row of p, A' moves to the left.
- If A moves up and it is at the top row of p, then A' skips two steps of A ahead, since A will return on the next step, by the assumptions.
- A' handles moves of A downward in a symmetrical way.
- If A moves left or right on the top row to state s', A' moves to the next or the last |-symbol, and continues the simulation from state s'.

Note that cell x being on the top row of p means that the corresponding cell of w either has | as a left neighbor, or it is the first cell of the word. Similarly, it is easy to check whether the automaton A being simulated is on the last row of p, which is needed in the case of moves downward. It is clear that the language of A' can be taken as L', since if $f(p) = w$, the runs of automaton A' on w are in exact correspondence with the runs of A on p in the sense explained previously, apart from the searches, which are deterministic, and never accept. Therefore $f(L) = L' \cap L_{pics}$, as required.

Conversely, let L be a regular language accepted by the one-way DFA A. We construct a CDFA A' accepting $f^{-1}(L)$: A' simply simulates a run of A by remembering the state in which A is at each point, and feeds this simulated copy of A the symbols on p in the top-down, left-to-right order.

Because f is a bijection, it follows from these characterizations that CDFA, RLG and CNFA are the exact same classes, since the f-images of their languages are exactly $L \cup L_{pics}$ where L ranges over the regular languages. □

Corollary 4.13. *$RLG \subsetneq DFA$.*

Proof. Clearly DFA contains languages RLG doesn't. For instance the language $L_{t=r} = \{p \in \Sigma_*^* \mid p[1,*] = p[*,|p|]^R\}$ can be implemented with a DFA that first checks that the picture is a square, and then, one cell of the top row at a time, moves diagonally to the rightmost column, checks that the symbol seen agrees with the one at the corresponding cell of the top row, and diagonally moves back on the top row.

An RLG for this language, on the other hand, would require the last stage 1 terminal of the first row to remember the whole contents of the first line. More precisely, let G be an RLG grammar for $L_{t=r}$. Then, if any of the languages L_d contain multiple words and can be used as the language of the last column, the same last column will be legal with multiple contents of the first row. This means only a finite amount of words can appear in the last column, which is a contradiction. □

In [1], it was asked if there is a connection between RLG and DREC. We prove a theorem that gives such a connection. Since the hard direction here is the inclusion of RLG in DREC, the characterization of RLG using XFA is not helpful.

Theorem 4.14. *$RLG \subsetneq DREC$*

Proof. The language $L_{t=r}$ can be implemented in DREC using diagonal signals, see Section 2.5, but it is not in RLG by the proof of the previous corollary. Therefore, DREC $\not\subseteq$ RLG.

For the other direction, let $G = (L, (L_d)_d)$ be an arbitrary normalized RLG grammar. For each L_d, let A_d be a deterministic one-dimensional one-way automaton with $\mathcal{L}(A_d) = L_d$. Given a picture p, on each column, we run all of A_d, feeding each of them the contents of the column. If the picture p is in $\mathcal{L}(G)$, then on the last row of the picture, we will necessarily have exactly one of the automata A_d in an accepting state, in each cell, since each column must be in exactly one of the disjoint languages L_d.

Now, it is enough to check, on each row, whether this is true, and if it is, whether the sequence of languages used for each column corresponds to a word of L. For this, let A be a deterministic one-way automaton recognizing L. On every row, we start a run of A, feeding it the unique d such that A_d is in an accepting state. The run fails if such a d does not exist. As final states, we can now choose the cells in which the run of A is accepting. □

Also the relations between REC, RE, SFRE and CFRE were asked in [HoFL]. The following sequence of theorems proves all the interesting relations. The crucial result here – that SFRE is not a subset of REC – has also been proved by Matz in [Matz], with a much simpler proof. Also the other direction was independently proved by him, years earlier, using on the same characterization of unary RE as we prove.

Definition 4.15. Given a picture $p \in L_{graphs}$, a clockwise rectangular cycle on p is a path z along the edges of the graph corresponding to p that has the following structure: z starts at the south node of some cell a, first moves $m_{\triangleright}$ steps to the right along west nodes reaching the west node of some cell b, then moves $m_{\triangledown}$ steps down along the north nodes of cells below b, reaching the north node of cell c, then moves $m_{\triangleleft} = m_{\triangleright}$ steps to the left along the east nodes of cells to the left of c, reaching some cell d and finally moves up from d until it reaches the south node of a.

The language L_{norecs} is the subset of pictures p of L_{graphs} such that there is no clockwise rectangular cycle in p.

Lemma 4.16. *The language L_{norecs} is in SFRE.*

Proof. Let us first construct a regular expression for L_{norecs}. We then explain how to remove the star operations from it. For a language L_{cycle} over Σ, consider

$$L_{nn} = \Sigma_*^* \ominus (\Sigma_*^* \oplus L_{cycle} \oplus \Sigma_*^*) \ominus \Sigma_*^*.$$

This expression simply surrounds a picture of L_{cycle} with an arbitrary picture over Σ.

By drawing a clockwise rectangular cycle along the edge of L_{cycle}, we obtain exactly the pictures in L_{norecs}^c with respect to alphabet Σ. Such a language L_{cycle} is given by the regular expression

$$\begin{array}{lllll} (\Sigma_{s\to e} & \oplus & \Sigma_{w\to e}^{*\oplus} & \oplus & \Sigma_{w\to s})^{\ominus} \\ (\Sigma_{s\to n}^{*\ominus} & \oplus & \Sigma_*^* & \oplus & \Sigma_{n\to s}^{*\ominus})^{\ominus} \\ (\Sigma_{e\to n} & \oplus & \Sigma_{e\to w}^{*\oplus} & \oplus & \Sigma_{n\to w}), \end{array}$$

where $\Sigma_{a\to b}$ is the set of symbols of Σ having a connection from the a-node (for instance, the west node if $a = w$) to the cell in direction b.

Therefore, L_{nn}^c is a regular expression for the language L_{norecs}. We are not done yet, however, since our regular expression contains star operations, which are not allowed in SFRE. Luckily, we only need the star operations for constructing full languages, full row languages, and full column languages, so by Lemma 4.6 and Lemma 4.7, there exists an SFRE for L_{norecs}. □

Lemma 4.17. *$L_{norecs} \notin REC$.*

Proof. We can directly use the construction of Theorem 2.63. Consider a hypothetical grammar G for the language L_{norecs}, and for a given n, the pictures $p_\prec$ corresponding to different total orders $\prec$ of $[1, n]$. The pictures $p_\prec$ do not contain cycles, and a fortiori do not contain rectangular cycles either. As in the proof of Theorem 2.63, the grammar G will necessarily allow the swapping of sides of two of these pictures p_1 and p_2. But the cycle obtained this way, in Theorem 2.63, is in fact rectangular, and since we know both of the two swaps of p_1 and p_2 are in L, one of the cycles obtained will rotate clockwise, proving $L_{norecs} \notin$ REC. □

We will need the following closure properties of the eventually periodic sets.

Lemma 4.18. *The eventually periodic sets are closed under boolean operations*

Proof. For complement, let X be an eventually periodic set with some blocks (A, B, C, D). Then the complement of X in $\mathbb{N}^2$ is just (A^c, B^c, C^c, D^c), where J^c denotes cellwise complement of the block J.

Let X_1 and X_2 be eventually periodic sets, and take common periods for the block representations (A_i, B_i, C_i, D_i) of their languages. Clearly, then, $X_1 \cap X_2$ and $X_1 \cup X_2$ are given by the blocks $(A_1, B_1, C_1, D_1) \cap (A_2, B_2, C_2, D_2)$ and $(A_1, B_1, C_1, D_1) \cup (A_2, B_2, C_2, D_2)$, respectively, where the operations are taken pointwise, in each block component. □

Even though RE can do rather complicated things, it is very weak when restricted to unary. This was also proved by Matz in [15]. Matz uses a slightly different definition, which we prove to be equivalent to ours in the following theorem. We find our definition to be more visually appealing.

Theorem 4.19. *The eventually periodic subsets of $\mathbb{N}^2$ are exactly the finite unions of cartesian products of eventually periodic one-dimensional sets.*

Proof. The eventually periodic sets are closed under the boolean operations, so we just need to prove that a cartesian product of eventually periodic one-dimensional sets is eventually periodic in $\mathbb{N}^2$. For this, simply note that once the threshold for the vertical one-dimensional set has been exceeded, every column becomes periodic, and the same is true for the rows. But this is exactly the definition of an eventually periodic set.

For the other direction, let X be an eventually periodic set with the block representation (A, B, C, D). Then for all $J \in \{A, B, C, D\}$, it is easy to construct a union of cartesian products of eventually periodic sets for the part of X covered using J-blocks: For A, this is the union of cartesian products of two singletons. For B and C, it is the union of cartesian products of singletons and periodic sets, and for D is it the union of cartesian products of periodic sets. □

Lemma 4.20. *The unary languages in RE are exactly the eventually periodic unary languages, that is, the unary languages L' such that the set $L = \mathit{shape}(L')$ is eventually periodic.*

Proof. We proceed by structural induction on RE expressions. First, consider a singleton language or the empty language $L' = \{p\}$ or $L' = \varnothing$. Certainly, the shapes of such languages are eventually periodic. We proceed by structural induction.

Boolean operations:

Direct from Lemma 4.18.

Concatenation:

Let r_1 and r_2 be regular expressions, and consider $r = r_1 \oplus r_2$. We show that $X = \mathrm{shape}(\mathcal{L}(r))$ has, for some t', q', the properties

$$w > t' \wedge (h, w) \in X \implies (h, w + q') \in X$$

and

$$h > t' \wedge (h, w) \in X \implies (h + q', w) \in X$$

which imply eventual periodicity by Lemma 3.59.

Let $X_i = \text{shape}(\mathcal{L}(r_i))$, and let t and q be common thresholds and periods of the languages X_i. Then, we choose $t' = 2t + 1$ and $q' = q$. Consider $(h, w) = (h, w_1 + w_2) \in X$ such that $w > t'$, where $(h, w_i) \in X_i$. For one of the i, we must have either $w_i > t$. Therefore, $(h, w_i + q) \in X_i$, and thus $(h, w + q') = (h, w_1 + w_2 + q) \in X$, as required. Now, let $(h, w) = (h, w_1 + w_2) \in X$ be such that $h > t'$, where $(h, w_i) \in X_i$. Then also $h > t$, so for both i, we have $(h, w_i) \in X_i \implies (h + q, w_i) \in X_i$, and thus $(h + q, w) = (h + q, w_1 + w_2) \in X$.

The case $r = r_1 \ominus r_2$ is symmetric.

Star operations:

Let r' be a regular expression, and consider $r = r'^{*\oplus}$. Again, we show that $X = \text{shape}(\mathcal{L}(r))$ has, for some t' and q', the properties

$$w > t \wedge (h, w) \in X \implies (h, w + q) \in X$$

and

$$h > t \wedge (h, w) \in X \implies (h + q, w) \in X$$

which imply eventual periodicity by Lemma 3.59.

Let $X' = \text{shape}(\mathcal{L}(r'))$. If $(h, w) \in X$, it is of the form $(h, \sum_i w_i)$, for some $(h, w_i) \in X'$. Thus if $h > t$, we have $(h + mq, \sum_i w_i) \in X$ for all natural numbers m.

As for the widths, let n be the product of all the smallest widths that occur with some h in X'. Such a number n exists, since the set of sets $W(h) = \{w \mid (h, w) \in X'\}$ is eventually periodic as h grows. Then, $(h, \sum_i w_i) \in X \implies (h, \sum_i w_i + n) \in X$ as long as $\sum_i w_i > 0$: At least one width w occurs with h in X', and thus if w' is the smallest element of $W(h)$, we may add n/w' copies of w' to $(h, \sum_i w_i)$ by the definition of the star operation. Therefore, mn can be taken as the horizontal period for any m, and thus $t' = t$ and $q' = nq$ can be taken as the threshold and period.

The case of $r^{*\ominus}$ is symmetric.

□

Theorem 4.21. *SFRE ⋈ REC and RE ⋈ REC.*

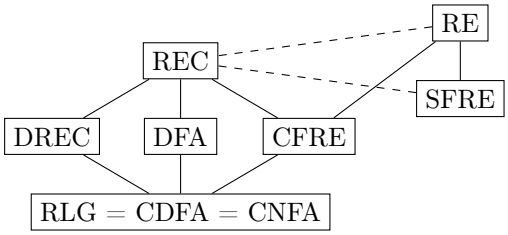
Proof. The square pictures belong to REC – RE by the previous lemma, since $\{(n, n) \mid n \in \mathbb{N}\}$ is clearly not eventually periodic, but was shown to be in REC in Chapter 2.1. From this is follows that REC ⊈ RE
On the other hand, REC ⊈ SREC, since the language L_{norecs} is in SFRE – REC by Lemma 4.16 and Lemma 4.17.

Since SFRE ⊂ RE, the claim follows from these. □

Finally, let us compare RLG to CFRE, and then summarize the results of this chapter in a Hasse diagram, depicted in Figure 12.

Theorem 4.22. *RLG ⊂ SRE ⊂ CFRE.*

Figure 12: Hasse diagram comparing the classes defined in this chapter with each other, and the classes from other chapters. Some uninteresting relations are omitted.



Proof. Let $G = (L, (L_d)_{d \in \Delta})$ be an RLG grammar. Let E be a one-dimensional regular expression for L, and let E' be the SRE picture expression obtained by changing all concatenation operations to the $\oplus$ operation. For each $d \in \Delta$, let E_d be a one-dimensional regular expressions for the language L_d, and let E'_d be the SRE picture expression obtained by changing all concatenation to the $\ominus$ operation. Then it is clear that if we substitute E'_d for d in E' for all $d \in \Delta$, we obtain an SRE picture expression for $\mathcal{L}(G)$. The latter claim follows from Theorem 4.4. □

5 Conclusions and future work

5.1 Conclusions

The main contribution of this thesis is the result that the classes NFA and FNFA coincide, that is, being able to leave the domain of a picture does not strengthen nondeterministic finite automata. This has been asked in at least [8], [9] and [7]. In order to prove this, we introduce the concept of *landing sequences*, which characterize all the ways an NFA can enter a region.

Other questions asked in the literature and solved here are the proper inclusions of AFA in FAFA and RLG in DREC. However, both results are rather straightforward to prove.

As for REC, we use a new approach for showing that the classes FO (first-order logic) and LTT coincide, namely that of not only defining the language of a first-order sentence, but also that of a first-order formula with free variables, by encoding variable information in the alphabet. Showing LTT includes FO then becomes a matter of proving the closure properties corresponding to variable introduction and quantification. The approach naturally generalizes to LTT and FO in other families of graphs.

In addition to building REC languages using the definition, and using its closure properties, we introduce the technique of recognizable relations. This allows the building of new languages using existing *constructions*. We show many applications of the technique. For finding languages *outside* REC, we prove a more general version of Matz' lemma. We show examples of languages outside REC for which the usual way of applying Matz' lemma does not work, and show how to modify the basic idea to work for them. We also completely characterize the neighborhoods that can be used to implement every REC grammar, slightly clarifying the results of [17].

Our approach to automata is somewhat unusual. We define only one kind of automaton, which works for all road-colored graphs. Automata with specific properties are then obtained by defining new graph representations of pictures. This unifies the main automata used in the theory of picture languages: automata restricted on the cells of the picture, automata that are allowed to exit the picture, automata that cannot *enter* the picture, and automata that only use a subset of the four cardinal directions. We also define a new automaton using this approach, the *comb automaton*, which turns out to characterize RLG.

We define the class UFA of languages of automata using only universal states. We note that certain arguments of [7] that build a proper chain out of the classes DFA, NFA and AFA can directly be adapted to build a diamond out of these classes and UFA. Again using the arguments of [7], we note that UFA is incomparable with REC. We also conjecture that this class is not closed under union.

We compare many pairs of classes and build small inclusion diagrams for the most interesting of these. In addition to the solutions to questions about relations asked in the literature, we in particular prove the inclusion of LTT in DREC. This appears to be rather straightforward, but the result has been omitted from at least [16], where LTT and DREC even appear in the same diagram.

We also briefly discuss purely language theoretical issues. Our classes of picture languages are defined as being closed under the renaming of symbols. We

explore the implications of this for defining classes by their closure properties. We also give an obvious sufficient condition for a class of languages to be its own second complement.

5.2 Future work

It seems that the class UFA we define in the case of picture languages is not quite as natural as its cousins DFA, NFA and AFA. In particular, we conjectured in Conjecture 3.17 that UFA is not closed under union. This question is interesting for two reasons. Firstly, practically every natural class is closed under this operation. Secondly, the class NFA not only has this property, but also has the property of being closed under intersection. Therefore, the result would prove an interesting asymmetry between two superficially symmetric classes.

The result that NFA and FNFA coincide raises the question of when, in general, does it help to be allowed to exit the finite region containing the picture, say when the languages are finite balls in an infinite group. It seems natural to assume that this is not rare, as NFA behave in a rather simple fashion on unary input. In particular, we make the following conjecture, with the obvious generalizations of the concepts to three dimensions.

Conjecture 5.1. *NFA = FNFA on three-dimensional pictures.*

Already here, our approach for the two-dimensional case does not directly work.

Connections between picture languages and the theory of two-dimensional shift spaces don't seem to have been researched much, although there certainly is much to be said. Due to lack of time and space, we could not include a chapter devoted to this.

We also hoped to compare, in this thesis, local grammar types by the REC languages that can be implemented using them as the underlying grammar. The universal local grammar types are at the top of this partially ordered set, and they have a very simple characterization (Theorem 2.19), but the general comparison problem seems to be nontrivial.

References

[1] D. Giammarresi and A. Restivo, Two-Dimensional Languages, in: A. Salomaa and G. Rozenberg, eds., Handbook of Formal Languages, Vol. 3 (Springer, Berlin, 1996)

[2] D. Giammarresi and A. Restivo, Recognizable Picture Languages, in: M. Nivat, A. Saoudi and P.S.P. Wang, eds., International Journal Pattern Recognition and Artificial Intelligence. Special issue on Parallel Image Processing, pp. 31-46, 1992

[3] K. Inoue and A. Nakamura, Some Properties of Two-Dimensional On-Line Tessellation Acceptors, in: Information Sciences, Vol. 13, pp. 95-121, 1977

[4] M. Blum and C. Hewitt, Automata on a Two-Dimensional Tape, in: IEEE Symposium on Switching and Automata Theory, pp. 155-160, 1967

[5] Oliver Matz, Regular Expressions and Context-Free Grammars for Picture Languages, in: Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science, pp. 283-294, February 27-March 01, 1997

[6] D. Giammarresi, A. Restivo, S. Seibert and W. Thomas, Monadic Second Order Logic over Rectangular Pictures and Recognizability by Tiling Systems, in: Information and computation, col. 125, no. 1, pp. 32-45, 1996

[7] J. Kari, C. Moore, New Results on Alternating and Non-Deterministic Two-Dimensional Finite-State Automata, in: A. Ferreira, H. Reichel, eds., 18th Ann. Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 2010, Springer, Berlin, 2001

[8] A. Rosenfeld, Picture Languages (Formal models for picture recognition), Academic press, New York, 1979

[9] K. Lindgren, C. Moore and M.G. Nordahl, Complexity of Two-Dimensional Patterns, in: Journal of Statistical Physics 91, pp. 909-951, 1998

[10] Michael Sipser, Halting Space-Bounded Computations, in: 19th Annual Symposium on Foundations of Computer Science (FOCS 1978), pp. 73-74, 1978

[11] Akira Ito, Katsushi Inoue and Itsuo Takanami, Deterministic Two-Dimensional On-Line Tessellation Acceptors are Equivalent to Two-Way Two-Dimensional Alternating Finite Automata Through 180°-Rotation, in: Theoretical Computer Science, Volume 66, Issue 3, pp. 273-287, 1989

[12] Tao Jiang, Oscar H. Ibarra and Hui Wang, Some Sesults Concerning 2-D On-Line Tessellation Acceptors and 2-D Alternating Finite Automata, in: Theoretical Computer Science, Volume 125, Issue 2, pp. 243-257, 1994

[13] Oliver Matz, On Piecewise Testable, Starfree, and Recognizable Picture Languages, in: Lecture Notes in Computer Science, Volume 1378/1998, pp. 203-210, 1998

[14] Mikołaj Bojańczyk, Thomas Colcombet, Tree-Walking Automata Cannot Be Determinized, in: Theoretical Computer Science - Automata, languages and programming: Logic and semantics (ICALP-B 2004) archive, Volume 350 Issue 2, 7 February 2006

[15] Oliver Matz, Classification of Picture Languages with Rational Expressions, Grammars and Logic Formulas, Diploma thesis (in German), 1996

[16] Alessandra Cherubini and Matteo Pradella, Picture Languages: From Wang Tiles to 2D Grammars, in: Lecture Notes in Computer Science, Volume 5725/2009, pp. 13-46, 2009

[17] Michel Latteux, David Simplot, Recognizable Picture Languages and Domino Tiling, in: Theoretical Computer Science archive, Volume 178 Issue 1-2, 1997

[18] Giusi Castiglione, Roberto Vaglica, Recognizable Picture Languages and Polyominoes, in: Lecture Notes in Computer Science, Volume 4728/2007, pp. 160-171, 2007

[19] Rohit J. Parikh, On Context-Free languages, in: Journal of the ACM (JACM), Volume 13 Issue 4, Oct. 1966

[20] Walter J. Savitch, Abstract Machines and Grammars, Little, Brown and Company, 1982, p. 49

[21] J. R. Büchi, On a Decision Method in Restricted Second-Order Arithmetic, in: 1960 Int. Congr. for Logic, Methodology and Philosophy of Science, pages 1-11, Stanford, SIAM-AMS, Stanford university press, 1962

A Estimates and calculations

Lemma A.1. *For any a, and large enough n,*

$$\binom{2^n}{2^{n-1}} > a^n.$$

Proof.

$$\binom{2^n}{2^{n-1}} = \frac{2^n(2^n-1)\cdots(2^{n-1}+1)}{2^{n-1}(2^{n-1}-1)\cdots 1} > 2^{2^{n-1}} > a^n,$$

because obviously

$$\log 2^{2^{n-1}} = 2^{n-1} > n \log a = \log a^n$$

for large enough n. □

Lemma A.2. *For any a, and large enough n*

$$2^{n^2} > a^n.$$

Proof. Obviously

$$\log 2^{n^2} = n^2 > n \log a = \log a^n$$

for large enough n. □

B Index of picture classes

Turku Centre *for* Computer Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www. tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Division for Natural Sciences and Technology
- Department of Information Technologies

ISBN 978-952-12-2584-0
ISSN 1239-1891