Technical Report 61

Church-Rosser Picture Languages and Their Applications in Picture Recognition

Hartmut Messerschmidt
Martin Stommel

TZI, Universität Bremen

TZI-Bericht Nr. 61
2011

TZI-Berichte

Herausgeber:
Technologie-Zentrum Informatik und Informationstechnik
Universität Bremen
Am Fallturm 1
28359 Bremen
Telefon: +49 421 218 94090
Fax: +49 421 218 94095
E-Mail: hq@tzi.de
http://www.tzi.de

ISSN 1613-3773

Church-Rosser Picture Languages and Their Applications in Picture Recognition

Hartmut Messerschmidt and Martin Stommel
Center for Computation and Communication Technologies,
Universität Bremen,
28359 Bremen, Germany
e-mail: {hmess,mstommel}@tzi.de

ABSTRACT

In image processing, there is a need for efficient methods that complement statistical models by structural information about the spatial scene arrangement and compositional hierarchy. In order to recognise the structure of locally detected features, we propose a two-dimensional Church-Rosser Picture Language that facilitates the evaluation of local information compared to one-dimensional languages. Although Church-Rosser languages are able to represent certain types of context-sensitivity, the word problem is solvable in linear time. We describe how the concept of local replacements used in rewriting systems and restarting automata, helps in pattern and picture recognition. It is shown that the Church-Rosser Picture Language can be recognised by a deterministic shrinking two-dimensional restarting automaton. The practical application of the Church-Rosser Picture Language in object recognition is illustrated.

Keywords: Church-Rosser Languages, Picture Recognition, Restarting Automata, Locality, Picture Languages

1. Introduction

In image processing the use of formal models with defined properties simplifies to a certain extend the interpretation of algorithms and mechanisms which are often characterised by heuristics and hidden approximations.

There are different approaches to use methods from formal languages to recognise pattern or pictures. The first approaches date back to the 1960s and 1970s [46, 7]. The used models range from context free to context sensitive grammars. While the first ones are not expressive enough to represent relevant visual patterns, the latter cannot be used in practice. Often the goal is to find extensions of context free languages that achieve a balance between complexity and expressive power.

In this paper, we address three particular problems of context free grammars: First, a grammar generates words of a language instead of accepting them. Secondly, context free grammars are nondeterministic, and third, the word problem of context free grammars is quadratic, extensions are even worse.

To overcome these drawbacks, the use of the not so widely known class of Church-Rosser languages (CRL) is proposed. On the one hand, Church-Rosser languages have a high expressive power, as many of the non context free languages that are considered important in pattern recognition are in fact Church-Rosser languages, e.g.

cross references and repetitions [23]. On the other hand, the word problem is solvable in linear time and there are deterministic automata models for this language class.

The application to image processing allows for the modeling of structural image information additionally to statistical information. While statistical information is useful during training to identify alphabets of selective patterns, the additional structural information allows for the evaluation of geometrical arrangements and hierarchical compositions [42, 52]. The benefit of Church-Rosser languages therefore consists in the efficient evaluation of arrangements of already detected local features.

To take full advantage of spatial information, Church-Rosser languages are extended to two-dimensional picture languages. Instead of extending the formal model, we take the less complex Church-Rosser languages and extend the dimensionality. Because reducing spatial information to one dimension destroys the neighbourhood of, and therefore the connection between, features.

In order to deterministically accept Church-Rosser picture languages, we introduce two-dimensional restarting automata and show that Church-Rosser picture languages are a subclass of the class of languages accepted by two-dimensional restarting automata. Compared to other methods for the pooling of spatially distributed information a rewriting system has the advantage of low-dimensionality. Voting mechanisms based on the Hough transform for example introduce one dimension for every free model parameter including position and pose[18].

This paper is structured as follows: The first sections review what is known about structural image information for object recognition (Sec. 2) and how automata and grammars are used to model this information (Sec. 3). In this context, two-dimensional variants of Church-Rosser languages promise the recognition of context-sensitive patterns at low computational complexity preserving image locality (Sec. 4). After the introduction of one-dimensional rewriting systems and Church-Rosser languages (Sec. 5) and a short review of related approaches for picture languages (Sec. 6), our approach is elaborated (Sec. 7) in more detail: A two dimensional rewriting system is introduced where local image sections can be replaced by weight-reduced and length-reduced sub-pictures. Local conditions are given that assert the connectivity of the pictures for a general class of replacement rules. Church-Rosser Picture Languages are introduced as a special case. It is then shown that all Church-Rosser Picture Languages can be accepted by a corresponding two-dimensional restarting automata. The remaining sections deal with the application of Church-Rosser Picture Languages in image processing. It is demonstrated that the method is able to count connected components (Sec. 8) and simulate sliding window filters (Sec. 9). It is outlined how Church-Rosser Picture Languages can be used for picture recognition (Sec. 10). A short summary concludes the paper.

2. Properties of Structural Information in Images

In the field of image processing, the use of language models is studied in the context of structural pattern recognition as opposed to the competing direction known as sta-

tistical pattern recognition. Structural pattern recognition focuses on the modeling of parts of objects and their mutual relationship. In statistical pattern recognition on the other hand, objects are modeled as feature vectors that refer to the whole object. These feature vectors are then classified by standard algorithms from discriminant analysis. Recent progresses in kernel-based methods [8] lead to impressive results in object recognition [16, 17, 55], and as a consequence statistical approaches predominate in current computer vision conferences.

However, it is also believed "that purely statistical methods have a natural bound, that can only be overcome by a close use of the structural nature of the data" [11]. There are thus many attempts [26, 58, 32] to complement statistical data by structural information.

The motivation to use compositional models has partly historical reasons. Marr reports [30] that the first technical approaches to object recognition had strong psychological and neurophysiological roots. A common disappointment [34] about early connectionist systems [45] directed research towards the step-wise construction of 3d-models from two-dimensional image data. Failures in these approaches identified noise and illumination variances as major problems in image processing [36, 29, 6].

Today, many general-purpose object recognition systems are based on features descriptors that model local image regions based on the gradient orientation [29, 2, 22, 33]. The advantage of such features is that they are largely invariant to illumination changes, geometric deformations and changes in viewpoint. A popular approach in statistical pattern recognition for example is to classify histograms of local features.

The use of local features is justified in different ways. Ommer et al. [42] mention robustness against missing parts together with compositionality as a general principle [14] in cognition and brain organization. Aycinena et al. [1] emphasise efficiency in the sense that parts can be re-used in different contexts. The importance of locality in part formation has been shown statistically [52].

The structure of objects is usually expressed in terms of geometrical relationships between parts. Since the structure of an object is often closely related to its function [47], structural information can be important for object recognition and image analysis.

The extend to which structural information is modeled is a trade-off between the accuracy of the model, numerical stability and computational effort. Crandall et al. [10, 9] show that the introduction of statistical part dependencies to a bag-of-features-model indeed increases the accuracy of the model. Interestingly, the connection of all parts of a constellation to a central reference part in a star-shaped manner was found almost optimal, while further connections did not improve the accuracy significantly. Fergus et al. [12] report that a star-shaped model outperforms a fully connected model both with respect to accuracy as well as speed. They argue that the sparse model is less prone to overfitting. Moreover, the sparse representation has only polynomial complexity as opposed to exponential in the fully connected case. In practice, the sparse model therefore allows to distinguish more than just six parts.

The importance of geometry also depends on the scale [51]. While geometry makes up around 50 per cent of the information on the level of small parts, a bag-of-features model may be appropriate for the modeling of whole objects.

3. Modeling of Structural Image Information by Grammars and Automata

Attempts to find a grammar or automata representation that corresponds to the understanding of images and objects as a hierarchy of robustly detectable parts in geometric relationships date back to the 1960s [46] and the influential works of Fu [7] in the 1970s.

Fu [13, 7] proposed a tree grammar where the terminals represent single grey-level pixels and the rule set describes expansions of non-terminals into trees of terminals and non-terminals that correspond to parts of images and starting points for further expansion of the image in the 2D-plane. While the problem of noise has already been recognised in early work [7], subsequent work identified run-time complexity, expressiveness and inference of the grammar model as difficulties.

Noise is usually handled by error-correcting grammars and stochastic expansions, which allows for experiments on natural images [7, 19, 1, 28] of rigid [1] or deformable objects [53]. Heuristic methods are used to obtain compact visual alphabets [32, 53, 57]. It is not uncommon to use manually predefined primitives [19, 28], as well. The ordering and grouping of visual elements in the image plane has a high impact on the run-time complexity. Liang et al. [27] propose a linearisation method as a possible solution. Such methods lead however to a partial or full loss of locality. Siskind et al. [49] study the uniqueness of models for the geometrical scene arrangement.

The relationship between parts is modeled in different ways. Tree grammars include the spatial relationship in the rule structure [7]. They cannot display two-dimensional distances correctly because of frequent one-dimensional concatenations and the resulting limited access to neighbouring elements. Han and Zhu [19] define constraints in an attributed grammar. Lin et al. [28] use a stochastic graph grammar which models logical expressions.

Although context sensitivity is claimed crucial for pattern recognition by some authors [54, 23], most approaches are limited to context free languages in order to achieve a practical run-time behavior. To overcome the limitations these are often expanded to deal with specific problems. Considering the experiments by Crandall et al. [10] on the part connectivity, this might be a valid approximation of the true structural image properties, at least for a certain broad class of object recognition problems. Less complex descriptions may be valid as well. Tanaka argues that in practical applications both images as well as sets of prototypes are finite, so a regular grammar should be sufficient [54]. A more expressive grammar however allows for more compact and more intuitive models.

4. Two-Dimensional Church-Rosser Languages for Picture Recognition

We propose to use a two-dimensional extension of Church-Rosser languages to model structural image data. While Church-Rosser languages combine a high expressiveness of the model with low computational complexity, the two-dimensional extension solves some of the traditional problems in the application to visual data. A restarting automata is proposed to accept the language in an Analysis by Reduction manner.

Kiefer and Schlieder [23] outline three problems from a pattern recognition application and propose mildly context sensitive languages as a solution. The languages include deterministic context free languages such as the language of correctly nested brackets (Dyck language), as well as mildly context sensitive languages such as cross references ($a^n b^m c^n d^m$) and repetitions ($a^n b^n c^n$). The latter are known not to be context free. Nevertheless the word problem for all these languages can be solved in linear time, as all these languages are Church-Rosser. Aside from the well known Chomsky-Hierachy, Church-Rosser languages therefore seem to be a good trade-off between expressiveness and complexity.

It is not easy to extend languages of words to "picture"-languages. Formal language models are very limited in generating or accepting pictures-languages. So why use two-dimensional Church-Rosser languages, when there is already an advantage in using (the traditional one-dimensional) Church-Rosser languages in pattern recognition or picture classification?

While the complexity advantage of Church-Rosser is obvious from theory, the extension to a two-dimensional picture language is justified by the two-dimensional nature of visual data.

Let us for example assume a face detection task, where a set of visual primitives like the eyes, nose and mouth has already been detected. The detections can be stored in a two-dimensional feature image that gives the number of a detected primitive for every pixel coordinate (cf. Fig. 4). A rewriting system would reduce words (i.e. combinations of visual primitives) locally until the membership of a pattern to the modeled class is determined or rejected. For a rewriting step to be conducted, two conditions arise from the image recognition application: First, a sufficient number of primitives needs to be present, and secondly, the primitives must occur in a valid spatial arrangement.

Although the modeling of two-dimensional structures in sequences on a one-dimensional tape is possible and leads to simpler automata definitions, it is comparatively inefficient and possible optimisations that arise from known properties of the visual input data are ignored.

Nondeterministically it is easy to find local structures, by just guessing which structure will be present and by guessing where it will be. In order to find a local structure deterministically this is not that easy. The pattern can be anywhere in the sequence and a full scan of the tape is needed to check if all needed parts are present and in the correct location. To make it even worse, this must be done for all possible patterns.

In picture recognition there are often false positives, that is a part of the picture is falsely classified as, for example, an eye. This may lead to many possible eyes and it is not practical to check for all eyes if there is a nose and a second eye nearby. In a two-dimensional structure there is no such problem, because in the current window the surroundings of the patterns can be seen.

5. Church-Rosser Languages

A language according to Narendran and Otto [31] is called Church-Rosser if it consists of all ancestors of a special symbol with respect to a finite, length reducing, and confluent string rewriting system.

Definition *Let Σ be a finite alphabet. A* string-rewriting system *R on Σ is a subset of $\Sigma^* \times \Sigma^*$. It induces several binary relations on Σ^*:*

- *The* single-step reduction relation $\rightarrow_R := \{ (u\ell v, urv) \mid u, v \in \Sigma^*, (\ell \rightarrow r) \in R \}$ *is the most basic of these.*
- *The* reduction relation $\rightarrow_R^*$ *induced by R is the reflexive and transitive closure of $\rightarrow_R$.*

If $u \rightarrow_R^* v$, then u is an ancestor *of v, and v is a* descendant *of u. If there is no $v \in \Sigma^*$ such that $u \rightarrow_R v$ holds, then the string u is called* irreducible *(mod R). By* $\mathrm{IRR}(R)$ *we denote the set of all irreducible strings. If R is finite, then* $\mathrm{IRR}(R)$ *is obviously a regular language. The string-rewriting system R is called*

- length-reducing *if $|\ell| > |r|$ holds for each rule $(\ell \rightarrow r) \in R$,*
- confluent *if, for all $u, v, w \in \Sigma^*$, $u \rightarrow_R^* v$ and $u \rightarrow_R^* w$ imply that v and w have a common descendant.*

If a string-rewriting system R is length-reducing, then each reduction sequence starting with a string of length n has itself at most length n. If, in addition, R is confluent, then each string $w \in \Sigma^*$ has a unique irreducible descendant $\hat{w} \in \mathrm{IRR}(R)$, which can be computed from w in linear time (see e.g. [4]). This observation was one of the main reasons to introduce the Church-Rosser languages in [31, 35]. In contrast to the original definition we use a different notation that accounts for the following developments: In [41] it has been shown that shrinking and length reducing Church-Rosser languages are equally powerful. In [56] a normal form for shrinking Church-Rosser languages was obtained that restricts t_1 and t_2 to single symbols, namely ¢ and \$. In addition it can be shown that the border markers do not need to be consumed at all during a reduction. Therefore it is equivalent to define Church-Rosser languages in the following way.

Definition *A language $L \subseteq \Sigma^*$ is a* Church-Rosser language *if there exists an alphabet $\Gamma \supsetneq \Sigma$, a finite, weight reducing, confluent string-rewriting system R on Γ, two*

symbols $¢, \$ \in (\Gamma \setminus \Sigma) \cap \mathrm{IRR}(R)$*, and a symbol* $Y \in (\Gamma \setminus \Sigma) \cap \mathrm{IRR}(R)$ *such that, for all* $w \in \Sigma^*$*,* $¢w\$ \to_R^* ¢Y\$$ *if and only if* $w \in L$*.*

A Church-Rosser language system CRLS is then the 6-tuple $C = (\Gamma, \Sigma, R, ¢, \$, Y)$*.* $L(C)$ *denotes the language accepted by the CRLS* C*.*

Throughout this paper $L(C)$ denotes the languages accepted by the system or automaton C.

Definition *A* restarting automaton*,* RRWW*-automaton for short, is a one-tape machine that is described by an 8-tuple* $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$*, where* Q *is the finite set of states,* Σ *is the finite input alphabet,* Γ *is the finite tape alphabet containing* Σ*, the symbols* $¢, \$ \notin \Gamma$ *are markers for the left and right border of the work space, respectively,* $q_0 \in Q$ *is the initial state,* $k \geq 1$ *is the size of the* read/write window*, and*

$$\delta : Q \times \mathcal{PC}^{(k)} \to \mathfrak{P}((Q \times (\{\mathsf{MVR}, \mathsf{MVL}\} \cup \mathcal{PC}^{\leq (k-1)})) \cup \{\mathsf{Restart}, \mathsf{Accept}\})$$

is the transition relation.

The automaton M *is a* deterministic restarting automaton (det-RRWW) *if* δ *is a (partial) function.*

Here, $\mathfrak{P}(S)$ denotes the powerset of the set S. The term $\mathcal{PC}^{(k)}$ denotes the set of *possible contents* of the read/write window of M, where

$$\mathcal{PC}^{(i)} := (¢ \cdot \Gamma^{i-1}) \cup \Gamma^i \cup (\Gamma^{\leq i-1} \cdot \$) \cup (¢ \cdot \Gamma^{\leq i-2} \cdot \$) \quad (i \geq 1),$$

and

$$\Gamma^{\leq n} := \bigcup_{i=0}^{n} \Gamma^i \quad \text{and} \quad \mathcal{PC}^{\leq (k-1)} := \bigcup_{i=1}^{k-1} \mathcal{PC}^{(i)} \cup \{\varepsilon\}.$$

The transition relation describes four different types of transition steps:

(1.) A *move-right step* is of the form $(q', \mathsf{MVR}) \in \delta(q, u)$, where $q, q' \in Q$ and $u \in \mathcal{PC}^{(k)}$, $u \neq \$$. If M is in state q and sees the string u in its read/write window, then this move-right step causes M to shift the read/write window one position to the right and to enter state q'. However, if the contents u of the read/write window is only the symbol \$, then no shift to the right is possible.

(3.) A *rewrite step* is of the form $(q', v) \in \delta(q, u)$, where $q, q' \in Q$, $u \in \mathcal{PC}^{(k)}$, $u \neq \$$, and $v \in \mathcal{PC}^{\leq (k-1)}$ such that $|v| < |u|$. It causes M to replace the contents u of the read/write window by the string v, and to enter state q'. Further, the read/write window is placed immediately to the right of v. However, some additional restrictions apply in that the border markers ¢ and \$ must not disappear from the tape nor that new occurrences of these markers are created. Further,

the read/write window must not move across the right border marker \$, i.e. if the string u ends in \$, then so does the string v, and after performing the rewrite operation, the read/write window is placed on the \$-symbol.

(4.) A *restart step* is of the form $\mathsf{Restart} \in \delta(q, u)$, where $q \in Q$ and $u \in \mathcal{PC}^{(k)}$. It causes M to move its read/write window to the left end of the tape, so that the first symbol it sees is the left border marker $\mathfrak{c}$. Also, M re-enters the initial state q_0.

(5.) An *accept step* is of the form $\mathsf{Accept} \in \delta(q, u)$, where $q \in Q$ and $u \in \mathcal{PC}^{(k)}$. It causes M to halt and accept.

There are various restricted types of restarting automata. They are obtained by combining two types of restrictions:

(a) Restrictions on the movement of the read/write window (expressed by the first part of the class name):

- RR- denotes no restriction,
- R- means that each rewrite step is immediately followed by a restart.

(b) Restrictions on the rewrite-instructions (expressed by the second part of the class name):

- -WW denotes no restriction,
- -W means that no auxiliary symbols are available (that is, $\Gamma = \Sigma$),
- ε means that no auxiliary symbols are available and that each rewrite step is simply a deletion (that is, if the rewrite operation $u \to v$ occurs in the transition relation of M, then v is obtained from u by deleting some symbols).

Restarting automata are defined as length reducing automata, as they have to reduce the length of the tape in every rewrite step. A slightly different model is the shrinking restarting automaton, first introduced in [43], and then studied in [21]. These restarting automata are less restricted than length reducing restarting automata. Each rewrite step of a shrinking restarting automaton is required to be weight reducing according to a weight function ϕ.

The following characterizations have been established in [5, 37, 38, 41].

Theorem 1 [39, 40, 21] *A language is Church-Rosser if and only if it is accepted by a shrinking* (det-)RRWW*, if and only if it is accepted by a length-reducing* (det-)RRWW.

6. Picture Languages

The previous sections considered string rewriting systems and automata to accept words. These approaches are now transferred to picture languages.

Definition *[15] A two-dimensional string (or a* picture*) over* Σ *is a two-dimensional rectangular array of elements of* Σ*. The set of all pictures over* Σ *is denoted by* Σ^{**}*. A two-dimensional language (or a* picture language*) is a subset of* Σ^{**}*.*

Given a picture $p \in \Sigma^{**}$*, let* $l_1(p)$ *denote the number of rows and* $l_2(p)$ *denote the number of columns of* p*. The pair* $(l_1(p), l_2(p))$ *is called the* size *of the picture. The empty picture is the only picture of size* $(0,0)$ *and it will be denoted by* λ*. Pictures of size* $(0,n)$ *or* $(n,0)$ *where* $n > 0$ *are not defined. The set of all pictures of size* (n,m) *with* $n, m > 0$ *will be indicated by* $\Sigma^{n \times m}$*.*

Furthermore, if $1 \leq i \leq l_1(p)$ *and* $1 \leq j \leq l_2(p)$*,* $p(i,j)$*, or equivalently* $p_{i,j}$ *denotes the symbol in* p *with coordinates* (i,j)*.*

Definition *Let* p *be a picture of size* (m,n)*. A* block*, or a* sub-picture*, of* p *is a* picture p' *that is a sub-array of* p*. That is, if* (m',n') *is the size of* p'*, then* $m' < m$ *and* $n' < n$ *and there exist integers* h*,* k $(h \leq m - m'$*,* $k \leq n - n')$ *such that* $p'(i,j) = p(i+h, j+k)$ *for all* $0 \leq i < m'$ *and* $0 \leq j < n'$*.* (h,k) *are the* starting coordinates *of the sub-picture.*

Remark This definition contrasts the definition of a sub-picture in [15], where $p'(i,j) = p(i+h, j+k)$ for all $0 \leq i \leq m'$ and $0 \leq j \leq n'$ holds.

To accept or generate picture languages there are different automata and grammar systems. The most common are:

- Four Way Automata,
- Cellular Automata,
- Line and Column grammars, and
- Tree grammars.

Four way automata are finite automata that operate on pictures and therefore can move in four directions. They were first introduced by Blum and Hewitt [3]. They obviously lack the ability to detect local structures on different scales. Cellular automata change the contents of the cells in parallel and communicate with adjacent cells. In the general case they are inefficient at higher scales. There are different restricted models. The on-line tessellation automaton [20] moves information only from the upper left corner diagonally over the picture. The neighbourhood is therefore only accessible in the lower right direction.

Both tree grammars [13] and grammars with vertical and horizontal rules [48] generate pictures while considering the context only in one dimension, i.e. is the neighbourhood of a given symbol follows a line. These grammars do not preserve or detect local two dimensional structures.

7. Church-Rosser Picture Languages

The concept of two-dimensional rewriting systems is that a sub-picture of a picture is replaced by another sub-picture. A problem is that there are different methods to make a picture smaller, whereas a one-dimensional tape is shortened in a straight forward way. Rewrite rules can introduce holes if the rewritten part is smaller than the original sub-picture. But holes negatively affect the image recognition because they constrain access to the local image environment. The best way to preserve locality would be to topologically "glue" the remaining part of the picture to the replaced part. But this is not applicable because the automaton to process such a language is much to complicated.

Therefore the rows and columns of the picture are shortened. The remaining parts of the picture just fall to the rewritten part. A problem in doing this is that the resulting picture might not be a rectangle any more (even when starting with a rectangle and if all rewriting rules rewrite rectangles into other rectangles). This happens if the replaced block has a different size than the full picture or the inserted block. Therefore, during a rewriting process, a picture is successively replaced by subsets of pictures, so called *possible pictures*.

In order to reduce a picture by successive rewrite steps, the possible picture must consist of only one connected part of symbols of the input or tape alphabet. To detect local structures at different scales, no possible picture should contain holes.

Definition *A* possible picture p *over* Σ *is an array of symbols of elements of* $\Sigma \cup \varepsilon$*, where* $\varepsilon \notin \Sigma$ *holds. Each row and column must contain at least one symbol* $x \in \Sigma$*. Let* $l_1(p)$ *denote the number of rows and* $l_2(p)$ *denote the number of columns of* p*. The pair* $(l_1(p), l_2(p))$ *is called the* size *of the possible picture. Each symbol* $p(k,l) = \varepsilon$ *is adjacent to another* ε *or to the border of the possible picture. Two coordinates* (x_1, y_1) *and* (x_2, y_2) *are called* adjacent *when either* $|x_1 - x_2| = 1$ *and* $y_1 = y_2$ *or* $|y_1 - y_2| = 1$ *and* $x_1 = x_2$ *hold. They are called* diagonally adjacent *when* $|x_1 - x_2| = 1$ *and* $|y_1 - y_2| = 1$ *hold.*

All coordinates $p(k,l) = \varepsilon$ *are called* empty*, the non-empty part of a possible picture is called the* content*. The content must be connected, i.e. each pair of elements of the content is linked via a chain of (diagonally) adjacent non-empty elements.*

These conditions ensure that there is always only one connected part containing symbols of Σ in a possible picture.

Remark: A sub-picture p' of possible picture p is not necessarily a possible picture. In a sub-picture, some rows or columns of p' can be empty, i.e. they contain only the symbol ε. Also, the content does not need to be connected. The other condition for a possible picture is valid for all sub-pictures of possible pictures. A block of a possible picture is called a *possible block*.

A rewrite step replaces a sub-picture of a possible picture with another sub-picture. Formally it is divided into two steps: First the application of a replacement rule and

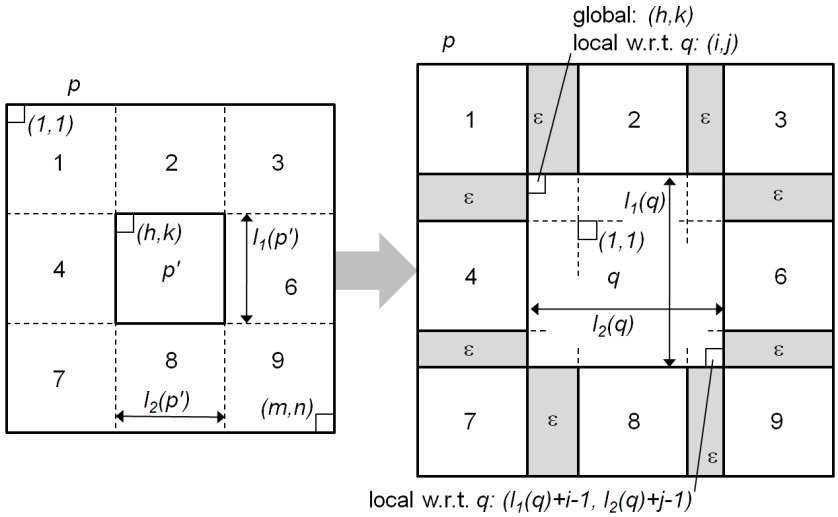


Figure 1: Illustration of rewriting the block p' inside p by the bigger block q.

second the consolidation of the picture.

Definition *An application of a replacement rule takes a block p' with starting coordinates (h,k) of a possible picture p of size (m,n) and replaces it with a possible block q (cf. fig. 1 for an illustration). p' has local coordinates ranging from $(1,1)$ to $(l_1(p'), l_2(p'))$. q can be larger than p' and its local coordinates can range from (i,j) to $(l_1(q)+i-1, l_2(q)+j-1)$ with $l_1(p') - l_1(q) \leq i \leq 1$ and $l_2(p') - l_2(q) \leq j \leq 1$. In each replacement $l_1(p') \leq l_1(q) + i - 1$ and $l_2(p') \leq l_2(q) + j - 1$ holds.*

The possible picture p can be concatenated from nine blocks: p' in the middle, block $1,2,3$ at the top from rows 1 to $h-1$ and columns 1 to $k-1$, k to $k+l_2(p')-1$, and $kl_2(p')$ to n, for block 1, 2, and 3, respectively. Blocks 4, $5 = p'$, and 6 are in the middle from rows h to $h+l_1(p')$ and blocks 7, 8 and 9 are at the bottom.

q is inserted into p in the following way: If j is smaller than 1, then $-j+1$ new columns are inserted in between column $k-1$ and k, that is right of block 1. Then block 2 follows, starting at column $k-j+1$. If $l_2(q)+j-1) - l_2(p')$ is greater zero, then this amount of new columns is inserted before block 3 follows.

If i is smaller than 1, then $-i+1$ new rows are inserted in between row $h-1$ and h. The content of this rows is only the content of the corresponding rows of q.

The next rows from $h-i+1$ to $h-i+1+l_1(p')$ contain block 4, the rows of q with local rows from 1 to $l_1)p')$ and block 6.

If $l_1(p') < l_1(q)+i-1$ holds, then new rows are inserted with the content of the rows $l_1(p')+1$ to $l_1(q)+i-1$ of q. The bottom contains blocks 7, 8, and 9 with possible gaps according to the ones between the blocks 1, 2, and 3.

A replacement rule is called non-increasing *if $i = j = 1$ and the size of p' equals the size of q.*

A replacement rule can reduce the size, even if its right hand side is larger than its left hand side. By enforcing $l_1(p') \leq l_1(q) + i - 1$ and $l_2(p') \leq l_2(q) + j - 1$, it is guaranteed that a certain row or column is deleted. Remark that this requirement does not restrict a replacement step.

For non-increasing replacement steps the definition can be expressed in a much shorter way: The block p' of p is replaced by q.

In the one-dimensional case, this rewrite of a sub-string in a word leads in a straight forward way to a binary relation on words. After a replacement, the result is not always a possible picture. In the so called *consolidation step* this composition is turned into a possible picture again. There are three different violations of a possible picture that might occur after a replacement step: First, there may be holes inside of the remaining picture, i.e. parts containing only the symbol ε. Secondly, complete rows or columns may be empty. Thirdly, the content may be disconnected. To overcome the second violation, empty rows and columns are deleted. It does not matter in which order this is done. Holes are filled up by adjacent symbols. Here it is important, in which ordering the rows and columns are falling to the rewritten part. There are different ways to do this, all with certain advantages and disadvantages. The two most promising ways for picture recognition are either that all sides fall in parallel to the empty parts, or that it is done in a defined ordering, here clockwise starting from right, bottom, left, top. The first method best preserves locality. It does however lead to "x"-shaped possible pictures and needs more computations. The second way is easy to perform and aligns the content to the upper left corner in a more compact arrangement. The disadvantage of the second one is that it might violate the locality more. Another method is to introduce special symbols which contain not only ε for an empty part, but also the direction from where it should be filled. All three ways might not fill all holes. If, after a consolidation step, holes remain in the composition, then it is rejected. Fortunately, it can be guaranteed with a local requirement for the replacement step that each hole can be filled.

There is another problem with filling holes in a picture after a rewrite step. Some parts might be disconnected by moving blocks. Any parts of the content that are not connected to the part where the rewrite step occurred (the *main part*), must be moved until they are connected. This is done as follows: Any now unconnected part was connected before the consolidation or became disconnected by the replacement rule. Therefore it is moved either in the same direction as the prior adjacent symbol until it is connected to the main part, or moved to the rewritten part.

If the content is disconnected because the right-hand side r of a replacement rule $l \to r$ has disconnected content and it does not become connected by moving blocks outside of r, then the composition is rejected. As a rule, parts of the right-hand side of a replacement rule are never moved in the following consolidation step.

To reconnect a composition, each symbol is moved in at most one direction. This

prevents intransparent shifting inside a possible picture.

Definition *A* consolidation step *transforms a composition after a replacement step back to a possible picture. First, each row and each column is deleted that contains only occurrences of the symbol ε. Second, any holes inside the content are filled. An occurrence of ε at the local position (h_l, k_l) inside the right hand side q of a replacement rule $p' \to q$ at position (h_g, k_g) does belong to three different cases:*

First none of the following conditions is true:

1. $q(h_l, j) = \varepsilon$ *for all* $k_l < j < l_2(q)$
2. $q(i, k_l) = \varepsilon$ *for all* $h_l < i < l_1(q)$
3. $q(h_l, j) = \varepsilon$ *for all* $0 < j < k_l$
4. $q(i, k_l) = \varepsilon$ *for all* $0 < i < h_l$

Then this coordinate remains empty. If this results in a hole in the content, then the possible picture is rejected.

Secondly, exactly one condition is true. Then the whole block of occurrences of ε is filled from this direction. If for example condition 1 is true, then the leftmost occurrence of ε that fulfils condition 1 in this row is taken and the block with starting coordinates $(h_g + l_1(q), k_g + l_2(q))$ and size $(l_1(p) - (h_g + l_1(q)), 1)$ is moved h places to the left, if (h, k_l) are the coordinates of the leftmost occurrence of ε. This move is possible because condition 1 states that the h places to the left of the moved block are empty. The other cases are handled accordingly.

Third, if, after all empty parts that only satisfy one condition are filled, there is an occurrence of ε that fulfils more than one condition, then the first satisfied condition is taken. When there is no block left that fulfils one of the conditions and the composition still contains holes, then it is rejected.

Finally any part of the content is moved that is not connected to the main part. It may be that any of the blocks $1, 3, 7, 9$ from the replacement step are unconnected. Block $1, 3, 7, 9$ are moved to down and right, down and left, up and right, and up and left until they are connected to other parts of the content, respectively. Then any part not connected to the main part is moved in the same direction as a previously adjacent symbol s that does now belong to the main part. This is done symbol by symbol and the top- and leftmost unconnected part is taken first.

The symbol s was moved either because one of the four conditions was fulfilled or because it was previously unconnected from the main part itself. If in the first case a part was connected by more than one symbol and these symbols were moved in different directions, then the one with the smallest numbered condition is taken. The movement stops if it is again connected to s, unless another symbol prevents the movement or a new hole would be created. The creation of a new hole can be prevented by moving on in the same direction, or, failing this, by stopping earlier. If this fails to connect the symbol with the main part, the possible picture is rejected.

Remark It does make a difference if first all blocks are moved and then the connectivity is checked, or if the connectivity is checked and restored after each step. The latter case might lead to new holes inside the content.

Definition *A* rewrite step *consists of the application of a replacement rule $l \to r$ at position (h,k) of a possible picture p, followed by a consolidation.*

Remark This detailed definition is needed to handle special cases. However, if the rewrite step is restricted to be for example non-increasing, then the replacement becomes much simpler. If only whole rows or columns contain occurrences of ε in the right hand side of a replacement rule, then the consolidation is much easier. In fact even with these strong restrictions, many problems in picture recognition can be solved and it is guaranteed that the locality of the picture is not strongly violated.

However, there is a weaker restriction that also guarantees a non-rejecting rewrite step.

Definition *A possible picture p is called* hassle-free *if its content is connected and for each empty symbol $p(h,k) = \varepsilon$ one of the following conditions holds:*

- $r(i,h) = \varepsilon$ *for all* $0 < i < k$
- $r(i,h) = \varepsilon$ *for all* $k < i < l_1(r)$
- $r(k,j) = \varepsilon$ *for all* $0 < j < h$
- $r(k,j) = \varepsilon$ *for all* $h < j < l_2(r)$

A replacement rule $l \to r$ is called hassle-free *if its right hand side r is hassle-free and for each coordinates (x,y) with $l(x,y) = \varepsilon$ it follows that $r(x,y) = \varepsilon$ holds, too. A rewrite step is called* hassle-free *if its replacement rule is.*

Then it can be shown that a hassle-free rewrite rule starting from a hassle-free possible picture is never rejected.

Definition *PP ist the set of all possible pictures. PB ist the set of all possible blocks. For $u \in PP$, $v, w \in PB$, the expression $v \subset u$ means that v is a sub-picture of u, and $u_{[v/w]^1}$ that exactly one occurrence of v in u is replaced by w by applying a rewrite step. So only if $v \subset u$ holds, $u_{[v/w]^1}$ can be constructed.*

A bordered picture $q = p(¢_1, ¢_2, \$_1, \$_2)$ is a possible picture of size $(l_1(p)+2, l_2(p)+2)$ where each row, except the top and bottom ones which contain only border markers, starts with a $¢_1$ and ends with a $\$_1$, and each column, except the left and right-most ones, which also contain only border markers, starts with a $¢_2$ and ends with a $\$_2$. p is a sub-picture of $p(¢_1, ¢_2, \$_1, \$_2)$ starting at coordinates $(2,2)$. The only empty parts of the bordered picture are the four corners having coordinates (1,1), $(1, l_2(p)+2)$, $(l_1(p)+2, 1)$ and $(l_1(p)+2, l_2(p)+2)$.

A bordered possible picture $p(¢_1, ¢_2, \$_1, \$_2)$ is a possible picture of size $(l_1(p)+2, l_2(p)+2)$ where each row, except the top and bottom ones, which contain only border markers,

starts with a $¢_1$ *and ends with a* $\$_1$*, and each column, except the left and right-most ones, which also contain only border markers, starts with a* $¢_2$ *and ends with a* $\$_2$*. p is a sub-picture of $p(¢_1, ¢_2, \$_1, \$_2)$ starting at coordinates $(2, 2)$.*

$\Sigma^{i,j}(¢_1, ¢_2, \$_1, \$_2)$ is the set of all bordered pictures $p(¢_1, ¢_2, \$_1, \$_2)$ where (i, j) is the size of p.

There are different ways to define a picture rewriting system. The probably most general is the following:

Definition *Let Σ be a finite alphabet. A* picture rewriting system *R on Σ is a subset of $PB \times PB$. Each element of R is a replacement rule. It induces several binary relations on PP:*

- *The* single-step reduction relation $\rightarrow_R := \{ (u, v) \mid u, v \in PP, (\ell \rightarrow r) \in R, \ell \subset u \text{ and } v = u_{[l/r]^1} \}$ *is the most basic of these.*
- *The* reduction relation $\rightarrow_R^*$ *induced by R is the reflexive and transitive closure of $\rightarrow_R$.*

If $u \rightarrow_R^* v$, then u is an ancestor *of v, and v is a* descendant *of u. If there is no $v \in PP$ such that $u \rightarrow_R v$ holds, then the possible picture u is called* irreducible *(mod R). By* $\mathrm{IRR}(R)$ *we denote the set of all irreducible possible pictures. The picture-rewriting system R is called*

- non-increasing *if $|l_1(\ell)| \geq |l_1(r)|$ and $|l_2(\ell)| \geq |l_2(r)|$ hold for each rule $(\ell \rightarrow r) \in R$,*
- size-reducing *if it is non-increasing and $|l_1(\ell)| > |l_1(r)|$ or $|l_2(\ell)| > |l_2(r)|$ holds for each rule $(\ell \rightarrow r) \in R$,*
- strictly-size-reducing *if $|l_1(\ell)| > |l_1(r)|$ and $|l_2(\ell)| > |l_2(r)|$ hold for each rule $(\ell \rightarrow r) \in R$,*
- reducing *if ℓ contains more non-empty symbols than r for each rule $(\ell \rightarrow r) \in R$,*
- strictly-reducing *if ℓ contains more non-empty symbols than r for each rule $(\ell \rightarrow r) \in R$ and R is non-increasing,*
- shrinking *if there exists a weight-function φ over Σ such that $\varphi(\ell) > \varphi(r)$ holds for each rule $(\ell \rightarrow r) \in R$,*
- strictly-shrinking *if there exists a weight-function φ over Σ such that $\varphi(\ell) > \varphi(r)$ holds for each rule $(\ell \rightarrow r) \in R$ and R is non-increasing,*
- confluent *if, for all $u, v, w \in PP$, $u \rightarrow_R^* v$ and $u \rightarrow_R^* w$ imply that v and w have a common descendant.*

Length-reducing Church-Rosser languages are as expressive as shrinking Church-Rosser languages. This is not known for Church-Rosser picture languages. Moreover, there are different ways to define length-reducing. Beside the above given definitions, it is possible to restrict a rewrite step to only reduce the number of symbols. On the

other hand, weight reducing rewrite steps might only perform rewrite steps that do not increase the width and height. Accordingly, a CRPL can be defined by using any of the above mentioned rewrite restrictions. Here we use the most general of them.

Definition *A language* $L \subseteq \Sigma^{++}$ *is a* Church-Rosser picture language (CRPL) *if there exists an alphabet* $\Gamma \supsetneq \Sigma$*, a finite, strictly-reducing, confluent picture-rewriting system* R *on* Γ*, four symbols* $¢_1, ¢_2, \$_1, \$_2 \in (\Gamma \setminus \Sigma)^* \cap \mathrm{IRR}(R)$*, and a symbol* $Y \in (\Gamma \setminus \Sigma) \cap \mathrm{IRR}(R)$ *such that, for all* $p \in PP$*,* $p(¢_1, ¢_2, \$_1, \$_2) \rightarrow_R^* Y(¢_1, ¢_2, \$_1, \$_2)$ *if and only if* $p \in L$*.*

A Church-Rosser picture language system CRPLS is an 8-tuple

$$C = (\Gamma, \Sigma, R, ¢_1, ¢_2, \$_1\$_2, Y).$$

If the rewriting system R *is not strictly reducing but shrinking, then the CRPL and the CRPLS is called shrinking. In general, a finite, confluent picture-rewriting system* R *with condition* $x \in \{$ *non-increasing, size-reducing, strictly-size-reducing, reducing, shrinking, strictly shrinking* $\}$ *is called a x-CRPLS.*

Definition *A* two-dimensional restarting automaton, 2D-RRWW-automaton *for short, is a one-space machine that is described by an 12-tuple* $M = (Q, \Sigma, \Gamma, ¢_1, ¢_2, \$_1, \$_2, \varepsilon, q_0, k, l, \delta)$*, where* Q *is the finite set of states,* Σ *is the finite input alphabet,* Γ *is the finite tape alphabet containing* Σ*, the symbols* $¢_1, ¢_2, \$_1$*, and* $\$_2 \notin \Gamma$ *are markers for the left, upper, right and bottom border of the work space, respectively,* $\varepsilon \notin \Gamma$ *is special symbol to mark empty parts of the tape,* $q_0 \in Q$ *is the initial state,* $k, l \geq 1$ *are the sizes for the width and height of the* read/write-window*, and*

$$\delta : Q \times \mathcal{PC}^{(k,l)} \rightarrow \mathfrak{P}((Q \times (\{\mathsf{MVR}, \mathsf{MVD}, \mathsf{MVUP}\} \cup \mathcal{PC}^{\leq(k),\leq(l)}) \cup \{\mathsf{Restart}, \mathsf{Accept}\})$$

is the transition relation.

The automaton M *is a* deterministic two-dimensional restarting automaton (2D-det-RRWW) *if* δ *is a (partial) function.*

Here $\mathfrak{P}(S)$ *denotes the powerset of the set* S*,* $\mathcal{PC}^{(k,l)}$ *is the set of* possible contents *of the read/write window of* M*, where* $\mathcal{PC}^{(i,j)}$ *are all possible blocks* q *of size at most* (i, j) *of a bordered possible picture* $p(¢_1, ¢_2, \$_1, \$_2)$*. If* $l_1(q) < i$ *holds, than the bottom row of* q *consists only of occurrences of the border marker* $\$_2$ *and If* $l_2(q) < j$ *holds, than the rightmost column of* q *consists only of occurrences of the border marker* $\$_1$*.* $\mathcal{PC}^{\leq(k),\leq(l)}$ *are all blocks of size at most* (i, j)*.*

Here the possible content of the read/write-window needs two parameters, the width and height. The restarting automaton is allowed to perform up and down movements to ensure that it can completely scan the picture in one cycle.

The transition relation describes five different types of transition steps:

(1.) A *move-right step* is of the form $(q', \mathsf{MVR}) \in \delta(q, u)$, where $q, q' \in Q$ and $u \in \mathcal{PC}^{(k,l)}$, $u \neq (\$_1, \cdots, \$_1)^T$. If M is in state q and sees the block u in its read/write-window, then this move-right step causes M to shift the read/write-window one position to the right and to enter state q'. However, if the contents u of the read/write-window consists only of occurrences of the symbol $\$_1$, then no shift to the right is possible.

(2.) A *move-down step* is of the form $(q', \mathsf{MVD}) \in \delta(q, u)$, where $q, q' \in Q$ and $u \in \mathcal{PC}^{(k,l)}$, $u \neq (\$_2, \cdots, \$_2)$. It causes M to shift the read/write-window one position to the bottom and to enter state q'. This, however, is only possible if the window is not already at the bottom of the picture.

(3.) A *move-up step* is of the form $(q', \mathsf{MVUP}) \in \delta(q, u)$, where $q, q' \in Q$ and $u \in \mathcal{PC}^{(k,l)}$, $u \neq (¢_2, \cdots, ¢_2)$. It causes M to shift the read/write-window one position up and to enter state q'. This, however, is only possible if the window is not already at the top of the picture.

(4.) A *rewrite step* is of the form $(q', v) \in \delta(q, u)$, where $q, q' \in Q$, $u \in \mathcal{PC}^{(k,l)}$, u does not contain only border markers, and $v \in \mathcal{PC}^{\leq(k),\leq(l)}$ such that v contains less symbols than u. It causes M to replace the contents u of the read/write-window by the possible picture v, and to enter state q'. Further, the read/write-window is placed immediately to the right of v. However, some additional restrictions apply in that the border markers must not disappear from the picture nor that new occurrences of these markers are created.

For one-dimensional rewrite steps, this condition was to ensure that always two border markers mark the ends of the tape. The number of border markers of a picture is not fixed but increases with the size of the picture. Only in one case, border markers are deleted from the picture. If a row or column is empty, i.e. it is $(¢_1\varepsilon^*\$_1)$ or $(¢_2\varepsilon^*\$_2)^T$, then this row or column is deleted completely from the picture.

Also, the read/write-window must not move across the right border marker $\$_1$, that is if each row of u ends in $\$_1$, then so does the possible picture v. After performing the rewrite operation, the read/write-window is placed on the block $(\$_1^k)^T$.

(5.) A *restart step* is of the form $\mathsf{Restart} \in \delta(q, u)$, where $q \in Q$ and $u \in \mathcal{PC}^{(k,l)}$. It causes M to move its read/write-window to the left and upper end of the picture, so that the top left symbol is empty, the leftmost non-empty symbol it sees is the left border marker $¢_1$ in each row and the topmost non-empty symbol is the upper border marker $¢_2$ in each column, and to re-enter the initial state q_0.

(6.) An *accept step* is of the form $\mathsf{Accept} \in \delta(q, u)$, where $q \in Q$ and $u \in \mathcal{PC}^{(k,l)}$. It causes M to halt and accept.

For two-dimensional restarting automata we have the same restrictions as for the one-dimensional ones.

(a) Restrictions on the movement of the read/write window (expressed by the first part of the class name):

RR- denotes no restriction,

R- means that each rewrite step is immediately followed by a restart.

(b) Restrictions on the rewrite-instructions (expressed by the second part of the class name):

-WW denotes no restriction,

-W means that no auxiliary symbols are available (that is, $\Gamma = \Sigma$),

ε means that no auxiliary symbols are available and that each rewrite step is simply a deletion (that is, if the rewrite operation $u \to v$ occurs in the transition relation of M, then v is obtained from u by replacing some symbols by ε).

Definition *A* configuration *of a two-dimensional restarting automaton is of the form $\alpha q(\beta_1, \beta_2)$, where q is the current state, and $\alpha + (\beta_1, \beta_2)^T$ is the bordered possible picture that is currently processed. Here, α denotes the sub-picture left of the current window. This part cannot be visited again before a restart. The read/write-window is on the top left part of β_2. If α and β_1 are empty, then it is a* restarting configuration. *If in addition β_2 contains only symbols from Σ, then it is called an* initial configuration.

A cycle *is the part of a computation from one restarting configuration to the next. A cycle can be divided into three part: First, the picture is scanned from left to right. Here, arbitrary up and down movements are permitted. Then one rewrite step is performed. Thirdly, the part to the left of the rewrite step is scanned from left to right, before the cycle ends with a restart operation. Again in this third part arbitrary up and down movements are permitted.*

Definition *The rewrite step of two-dimensional restarting automata is defined as being strictly-reducing. If the rewrite step of a two-dimensional restarting automaton A is not strictly-reducing, but condition $x \in \{$ non-increasing, size-reducing, strictly-size-reducing, reducing, shrinking, strictly shrinking $\}$ holds, then it is called an $x-$two-dimensional restarting automaton.*

If the width or the height is increased during a rewrite step, then the picture might grow. In this case, not all columns or rows are framed by border markers. This problem can be solved by framing the newly created rows and columns by additional border markers. Definition and figure 1 explicitly state the new inserted rows and columns.

In the following theorem, a connection between Church-Rosser picture languages and two-dimensional restarting automata is shown.

Theorem 2 *For each x-CRPL L there exists a deterministic two-dimensional x-*RWW*-automaton R such that $L(R) = L$ holds. $x \in \{\varepsilon$, non-increasing, size-reducing, strictly-size-reducing, reducing, shrinking, strictly shrinking $\}$*

Proof. Let $x \in \{\varepsilon$, non-increasing, size-reducing, strictly-size-reducing, reducing, shrinking, strictly shrinking $\}$,

let C be a x-CRPLS . We will construct a two-dimensional x-RWW-automaton A with $L(C) = L(A)$.

Let $p(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$ be a bordered picture. If $p \in L(C)$ then there exists at least one sequence of rules from R starting from $p(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$ and ending in $Y(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$. And as R is confluent, each sequence starting from $p(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$ ends in $Y(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$. A will simulate one of these sequences by always performing the leftmost possible rewrite step.

A moves from top to bottom and from left to right over the picture. Whenever it encounters a possible rewrite step, it is executed.

If the sequence ends with $Y(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$, then A will do likewise and therefore accept. If, on the other hand, A does not end in $Y(\text{¢}_1, \text{¢}_2, \$_1, \$_2)$, then A rejects. But that means that C will not accept the word either.

Thus $L(C) = L(A)$ holds.

□

Unfortunately, it remains open whether the other direction holds or not. The proof idea in the one-dimensional case is that a Church-Rosser language system can save the states on the tape. Because it simulates a deterministic restarting automaton, it can keep this information up-to-date. This idea can not be transferred to the two-dimensional case. However we have the following strong conjecture:

Conjecture *For each stateless two-dimensional x-*RRWW*-automaton A there exists a x-CRPLS C such that $L(A) = L(C)$ holds.*

An automaton is called stateless if it does only have one internal state, for further information see [24, 25]. It is unknown whether stateless two-dimensional restarting automata are less expressive than their counterpart with states. Solving this does also solve the question, whether each language accepted by a two-dimensional RRWW-automaton is a Church-Rosser picture language.

8. Counting Connected Components

The application of the rewriting system will be briefly demonstrated in the counting of connected components which is a standard problem in many computer vision tasks (e.g. in character recognition). The meaning of this demonstration is that related methods like four-way automata, cellular automata and grammars cannot solve it because they disregard local connectivity. The problem has also been used to argue against Perceptrons [34]. Given a black and white image, a black connected component is recursively defined as a single black pixel and all other black pixels that are

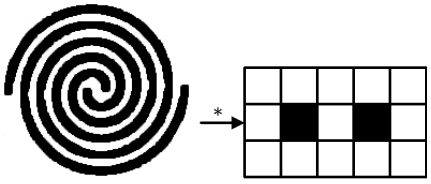
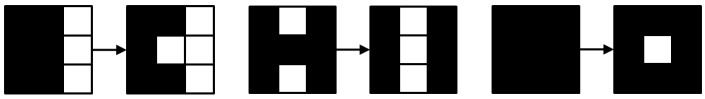


Figure 2: Counting connected components. Left: Two spiral shaped connected components. Right: The components have been reduced to separate, single pixels. Defining the weight of black pixels as 2 and the weight of white pixels as 1, the weight of the picture (minus its size) represents the number of connected components.



a) reduction b) separation c) piercing

Figure 3: Rules for counting connected components

adjacent to that connected component. For simplicity, we assume topological homeomorphism to a circular disk because it saves us lengthy descriptions of the labeling of different components. Figure 2 (left) shows an example of two connected components.

The approach for counting connected components is to reduce each component to a single pixel. Suitable weights are 2 for black pixels and 1 for white pixels. After reduction, the number of connected components is represented by the weight of the remaining symbols minus the image size (width $\times$ heigth).

To reduce the components, a set of rules is introduced that take a 3×3 block with a black center pixel and rewrite it by a 3×3 block with a white center pixel (cf. fig. 3a). Out of the 2^8 possible rules only those rules are created that preserve the connectivity of the border pixels of the block (fig. 3b shows a counter-example) and homeomorphism (fig. 3c shows a counter-example). The connectivity of the border pixels must be preserved to avoid the possible partitioning of a connected component into disconnected parts. Replacements in the border of a 3×3 block carry the danger of creating holes as well. These rules are sufficient to reduce connected components to single pixels. Please note that it is not necessary to use auxiliary symbols to solve the problem. After the rewriting process, the image consists of single black pixels on a white background. Since the weight of the black pixels is by 1 greater than the weight of the white pixels, the number of components is given by the weight of the image subtracted by its size.

9. Sliding Window Filters

Next, we will demonstrate that the picture language is consistent with the concept of sliding windows in image processing. The sliding window technique is a basic method of local filtering, where the pixel values of an output image depend on a local environment around each coordinate in the input image. First we need to separate an alphabet $\Sigma \subset \Gamma$ of the input image from the disjunct alphabet $\Phi \subset \Gamma, \Sigma \cap \Phi = \{\}$ of the output image. The contents of the input image stems from Σ, whereas the contents of the output image stems from Φ. The alphabets are necessary to avoid recursive filtering by separating input and output images. Phenomenologically, they may refer to the same colours.

The basic idea is to define rules of the kind

$$\begin{array}{lll} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{array} \rightarrow \begin{array}{lll} \diamondsuit & \diamondsuit & \diamondsuit \\ \diamondsuit & \diamondsuit & \diamondsuit \\ \diamondsuit & \diamondsuit & u \end{array}, \tag{1}$$

where $u = u(v_1, \ldots, v_9) \in \Phi$ is the filtering result, and the symbol $\diamondsuit$ indicates a *don't care element* on the left hand side of a rule or *not changing the original element* of the input image when used on the right hand side. It is customary to define rules for all possible input patterns $v_1, \ldots, v_9$ (each $v \in \Sigma$) for the filtering process to run smoothly. To achieve a defined sequence of execution from right to left and bottom to top, the lower right environment of the input pattern must be a corner, border, or element of the output alphabet. This can be achieved by defining rules such as

$$\begin{array}{llll} v_1 & v_2 & v_3 & \diamondsuit \\ v_4 & v_5 & v_6 & \diamondsuit \\ v_7 & v_8 & v_9 & \text{¢} \\ \diamondsuit & \diamondsuit & \text{¢} & \diamondsuit \end{array} \rightarrow \begin{array}{llll} \diamondsuit & \diamondsuit & \diamondsuit & \diamondsuit \\ \diamondsuit & \diamondsuit & \diamondsuit & \diamondsuit \\ \diamondsuit & \diamondsuit & u & \diamondsuit \\ \diamondsuit & \diamondsuit & \diamondsuit & \diamondsuit \end{array} \tag{2}$$

for the initial position of the sliding window at the lower right corner, or

$$\begin{array}{llll} v_1 & v_2 & v_3 & \diamondsuit \\ v_4 & v_5 & v_6 & \diamondsuit \\ v_7 & v_8 & v_9 & u' \end{array} \rightarrow \begin{array}{llll} \diamondsuit & \diamondsuit & \diamondsuit & \diamondsuit \\ \diamondsuit & \diamondsuit & \diamondsuit & \diamondsuit \\ \diamondsuit & \diamondsuit & u & \diamondsuit \end{array} \tag{3}$$

for the inner part of the image, where the element $u' \in \Phi$ from a previous filtering step must be from the output alphabet. It is obvious that this procedure in theory allows for the modeling of linear, rank order, and morphological operators. This includes a number of efficient existing implementations.

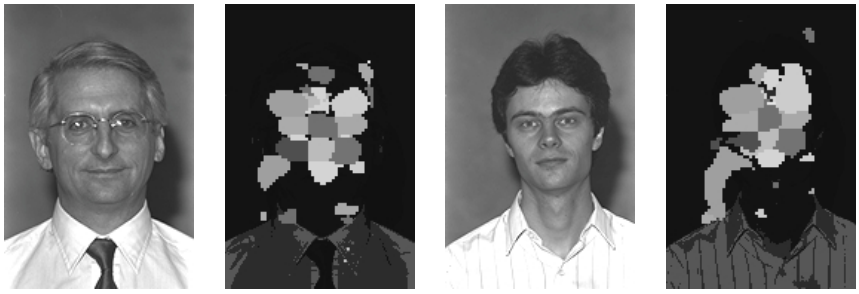


Figure 4: Original images [44] and detections of facial features [50] for two sample images. The detected areas of the eyes, brows, nose, nasal bone, mouth, cheeks are highlighted in different grey levels. For orientation, the outline of the face and shirt are indicated as well. A certain amount of misdetections is visible.

10. How to use Church-Rosser Picture Languages in Picture Recognition

Although this article primarily deals with complexity issues of the rewriting system, a few considerations will be given on the inference of the model.

One of the major problems in image processing is illumination invariance. In image processing, good results have been achieved by modeling the appearance of an object by local patterns of the gradient orientations. Therefore, a modeling of illumination changes by rewriting rules is not essential. Instead, we propose to use feature images as the starting point for the rewriting process. In a feature image, a label is assigned to every pixel coordinate. The label indicates the presence or absence of a certain feature, pattern or object. Figure 4 shows two feature images from a face recognition task [50] as an example.

The advantage of two-dimensional Church-Rosser languages is the possibility to access geometrical information. To preserve this information in the process of rewriting, the corresponding rules could define a uniform subsampling over the image plane. This is consistent with the compositional approach and shows correspondences to the concept of an image pyramid in image processing. Figure 5 gives an illustration.

To abet confluence, different levels of the pyramid should be modeled by separate visual alphabets. A training method would begin with the computation of the pyramids for a number of sample images. In a top-down process, the elements of the higher levels would be differentiated into constellations of elements on the lower levels. However, the high number of possible rule extensions demands a stronger generalisation over varying patterns. One way to achieve this is to introduce rewriting rules that simulate smoothing operations corresponding to e.g. certain morphological operators.

A strict scale factor between the pyramid levels limits the application of rewriting rules to small sets of neighbouring elements.

On the other hand, special symbols can be introduced to code large homogeneous regions. This way, irrelevant regions can be compressed very quickly. The computation

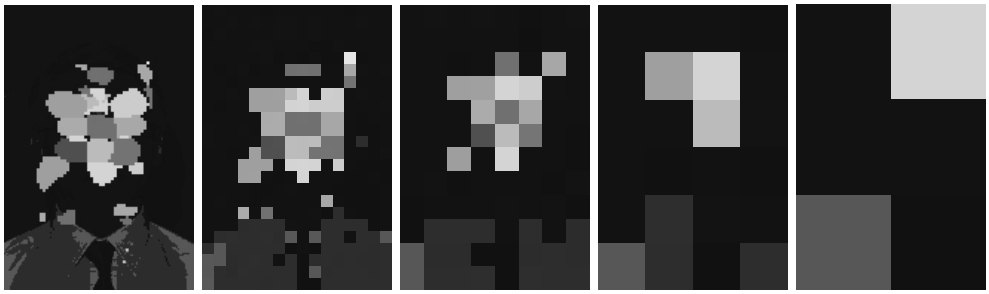


Figure 5: An image pyramid represents the feature arrangement at different sampling rates σ.

time is therefore spend on the difficult parts of the image. Here the search for local structures also helps in detecting noise, that is false classified features.

This can be best described by using the example of the face recognition (see figure 4). In both pictures there are singular features that are misdetections. These false positives can be detected by analyzing their surroundings. For example the small dots at the bottom of the left picture are certainly not part of a face. This allows for classification methods with a higher precision, that is more features are detected correctly. These methods often have a bad recall, that is they do lead to more false positives.

By starting a recognition process with the marking of these large homogeneous regions, deleting single features in it, and then combining regions that belong to the same feature, the picture is reduced very fast. And the more difficult computations take place on a much smaller picture.

The deletion of false positive is also useful, when the classification is allowed to label an area with more than one feature. Thus making it more likely that the area is labeled with the correct feature, but making the computation much more complicated.

At present, the possibilities of Church-Rosser languages for the modeling of visual data have not been clearly outlined. In contrast to a Perceptron [34], connected figures do not seem problematic for modeling. The hierarchical language structure seems also well suited to represent recursively embedded patterns over multiple scales. Such structures are more problematic in global approaches based on voting or feature constellations. Further research will show which visual patterns are consistent with Church-Rosser languages. From an empiric point of view, a set of design patterns for rewriting operations could also be useful to explore the domain of practical applications.

11. Conclusion

The modeling of structural information in images is highly relevant for many computer vision tasks. The analysis of spatial relationships between the parts of a scene provides much a more precise clues about the content of an image than the pure enumeration of depicted objects. Recent research results shows that the image structure in certain object recognition tasks corresponds well with context free languages. Context free languages are however inattractive because of their quadratic computational complexity. Often extensions of context free methods are used, which increases the complexity even more.

To overcome this computational limitation, we propose to use two-dimensional Church-Rosser languages. While Church-Rosser languages are sufficiently expressive for the modeling of complex visual information, they are only linear in run-time with respect to the number of pixels. Compared to the simpler case of one-dimensional languages, two-dimensional Church-Rosser languages have the advantage of preserving neighbourhood information. This avoids typical problems in traditional applications of formal languages for computer vision tasks.

While language models represent structural information in a generative way, automata models seem more appropriate to guide the object recognition procedure. Therefore, we introduce a deterministic shrinking two-dimensional restarting automaton. New transition relations extend movements within the input data from the one-dimensional case to two dimensions. Special care is taken to maintain connectivity and handle border-markers in the two-dimensional case. It is proven that every generalised Church-Rosser picture language can be transformed to an equivalent automaton.

A hypothetical application to an object recognition task identifies a number of image operators that are provided by the proposed rewriting system. The hierarchical language representation thereby seems well suited to model image pyramids to achieve scale-invariance. The rewriting of wide homogeneous regions in single symbols allows for an early refusal of incomplete matches. Smoothing operations allow for a good generalisation over varying object appearances.

References

[1] M. Aycinena, L. P. Kaelbling, T. Lozano-Perez, *Learning Grammatical Models for Object Recognition*. Technical Report MIT-CSAIL-TR-2008-011, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2008.

[2] H. Bay, A. Ess, T. Tuytelaars, L. van Gool, SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)* **110** (2006) 3, 346–359.

[3] M. Blum, C. Hewitt, Automata on a 2-Dimensional Tape. In: *FOCS*. 1967, 155–160.

[4] R. Book, F. Otto, *String-Rewriting Systems*. Springer-Verlag, New York, 1993.

[5] G. Buntrock, F. Otto, Growing context-sensitive languages and Church-Rosser languages. Information and Computation **141** (1998), 1–36.

[6] M. Burge, W. Burger, W. Mayr, Recognition and learning with polymorphic structural components. *Proc. of the 13th ICPR, Vienna, Austria* **1** (1996), 19–28.

[7] N.-S. Chang, K.-S. Fu, Parallel Parsing of Tree Languages for Syntactic Pattern Recognition. *Pattern Recognition* **11** (1979), 213–222.

[8] C. Cortes, V. Vapnik, Support-Vector Networks. *Machine Learning* **20** (1995) 3, 273–297.

[9] D. Crandall, D. Huttenlocher, Weakly Supervised Learning of Part-Based Spatial Models for Visual Object Recognition. In: *Proc. of European Conf. on Computer Vision (ECCV)*. 2006, 16–29.

[10] D. J. Crandall, P. F. Felzenszwalb, D. P. Huttenlocher, Spatial Priors for Part-Based Recognition Using Statistical Models. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1, 2005, 10–17.

[11] C. de la Higuera, A bibliographical study of grammatical inference. *Pattern Recognition* **38** (2005), 1332–1348.

[12] R. Fergus, P. Perona, A. Zisserman, A Sparse Object Category Model for Efficient Learning and Complete Recognition. In: J. Ponce, M. Hebert, C. Schmid, A. Zisserman (eds.), *Toward Category-Level Object Recognition*. LNCS 4170, Springer, 2006, 443–461.
`http://www.robots.ox.ac.uk/~vgg`

[13] K. S. Fu, *Syntactic Pattern Recognition and Applications*. Prentice-Hall, 1982.

[14] S. Geman, D. F. Potter, Z. Chi, Composition systems. *Quarterly of Applied Mathematics* **LX** (1998), 707–736.

[15] D. Giammarresi, F. Venezia, A. Restivo, Two-Dimensional Languages. 1997.

[16] K. Grauman, T. Darrell, The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. *IEEE International Conference on Computer Vision (ICCV)* **2** (2005), 1458–1465.

[17] K. Grauman, T. Darrell, Approximate Correspondences in High Dimensions. *In Advances in Neural Information Processing Systems (NIPS)* (2006).

[18] W. E. L. Grimson, D. P. Huttenlocher, *On the Sensitivity of the Hough Transform for Object Recognition*. Technical Report A.I. Memo No. 1044, Massachusetts Institute Of Technology, Artificial Intelligence Laboratory, 1988.

[19] F. Han, S. C. Zhu, Bottom-up/Top-Down Image Parsing by Attribute Graph Grammar. *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV)* **2** (2005), 1778–1785.

[20] K. Inoue, A. Nakamura, Some properties of two-dimensional on-line tessellation acceptors. *Inf. Sci.* **13** (1977) 2, 95–121.

[21] T. Jurdziński, F. Otto, Shrinking restarting automata. *International Journal of Foundations of Computer Science* **18** (2007), 361–385.

[22] Y. Ke, R. Sukthankar, PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. *Computer Vision and Pattern Recognition (CVPR)* **2** (2004), 506–513.

[23] P. Kiefer, C. Schlieder, Exploring Context-Sensitivity in Spatial Intention Recognition. In: B. Gottfried (ed.), *1st Workshop on Behaviour Monitoring and Interpretation (BMI'07)*. 296, CEURS Proceedings, 2007, 102–116.

[24] M. Kutrib, H. Messerschmidt, F. Otto, On stateless deterministic restarting automata. *Acta Inf.* **47** (2010) 7-8, 391–412.

[25] M. Kutrib, H. Messerschmidt, F. Otto, On Stateless Two-Pushdown Automata and Restarting Automata. *Int. J. Found. Comput. Sci.* **21** (2010) 5, 781–798.

[26] S. Lazebnik, C. Schmid, J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* **2** (2006), 2169–2178.

[27] P. Liang, M. Narasimhan, M. Shilman, P. Viola, Efficient Geometric Algorithms for Parsing in Two Dimensions. *International Conference on Document Analysis and Recognition (ICDAR)* (2005).

[28] L. Lin, T. Wu, J. Porway, Z. Xu, A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition* **38** (2009), 1297–1307.

[29] D. G. Lowe, Object recognition from local scale-invariant features. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 1999, 1150–1157.

[30] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and Company, 1982.

[31] R. McNaughton, P. Narendran, F. Otto, Church-Rosser Thue systems and formal languages. J. Assoc. Comput. Mach. **35** (1988), 324–344.

[32] K. Mikolajczyk, B. Leibe, B. Schiele, Multiple Object Class Detection with a Generative Model. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006.

[33] K. Mikolajczyk, C. Schmid, A Performance Evaluation of Local Descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* **27** (2005) 10, 1615–1630.

[34] M. L. Minsky, S. Papert, *Perceptrons: An introduction to computational geometry*. MIT Press, Cambridge, MA, USA, 1969.

[35] P. Narendran, F. Otto, K. Winklmann, The uniform conjugacy problem for finite Church-Rosser Thue systems is NP-complete. Information and Control **63** (1984), 58–66.

[36] S. Nayar, H. Murase, S. Nene, *Parametric appearance representation. In Early Visual Learning.*. Oxford University Press, 1996.

[37] G. Niemann, *Church-Rosser Languages and Related Classes*. Doktorarbeit, Fachbereich Mathematik/Informatik, Universität Kassel, 2002.

[38] G. Niemann, F. Otto, The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. In: M. Nivat (ed.), *Foundations of Software Science and Computation Structures, FoSSaCS'98, Proc.*. Lecture Notes in Computer Science 1378, Springer-Verlag, Berlin, 1998, 243–257.

[39] G. Niemann, F. Otto, Restarting automata, Church-Rosser languages, and representations of r.e. languages. In: G. Rozenberg, W. Thomas (eds.), *Developments in Language Theory - Foundations, Applications, and Perspectives, DLT 1999, Proc.*. World Scientific, Singapore, 2000, 103–114.

[40] G. Niemann, F. Otto, Further results on restarting automata. In: M. Ito, T. Imaoka (eds.), *Words, Languages and Combinatorics III, Proc.*. World Scientific, Singapore, 2003, 352–369.

[41] G. Niemann, F. Otto, The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. Information and Computation **197** (2005), 1–21.

[42] B. Ommer, J. M. Buhmann, Object Categorization by Compositional Graphical Models. *EMMCVPR'05, LNCS 3757* (2005), 103–113.

[43] F. Otto, T. Jurdziński, *On left-monotone restarting automata*. Mathematische Schriften Kassel 17/03, Universität Kassel, 2003.

[44] P. J. Phillips, P. J. Rauss, S. Z. Der, *FERET (Face Recognition Technology) Recognition Algorithm Development and Test Results*. Technical Report 995, Army Research Lab, 1996.

[45] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* **65** (1958) 6, 386–408.

[46] A. Rosenfeld, Picture Processing by Computer. *ACM Computing Surveys (CSUR)* **1** (1969) 3, 147–176.

[47] J. Schlecht, K. Barnard, E. Spriggs, B. Pryor, Inferring Grammar-based Structure Models from 3D Microscopy Data. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2007).

[48] G. Siromoney, R. Siromoney, K. Krithivasan, Picture languages with array rewriting rules. *Information and Control* **22** (1973) 5, 447 – 470. `http://www.sciencedirect.com/science/article/pii/S0019995873905731`

[49] J. M. Siskind, J. J. S. Jr., I. Pollak, M. P. Harper, C. A. Bouman, Spatial Random Tree Grammars for Modeling Hierarchical Structure in Images with Regions of Arbitrary Shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29** (2007) 9, 1504–1519.

[50] M. Stommel, O. Herzog, Learning of Face Components in Coherent and Disturbed Constellations. *Int'l Conf. on Image and Vision Computing New Zealand (IVCNZ), Queenstown, New Zealand, Nov. 8-9* (2010).

[51] M. Stommel, K.-D. Kuhnert, A Hierarchical Model for the Recognition of Deformable Objects. *In Proc. International Conference on Computer Vision and Graphics (ICCVG)* **Springer, LNCS 5337** (2008), 410–419.

[52] M. Stommel, K.-D. Kuhnert, Part Aggregation in a Compositional Model based on the Evaluation of Feature Cooccurrence Statistics. *Int'l Conf. on Image and Vision Computing New Zealand (IVCNZ), Christchurch, New Zealand, Nov. 26-29* (2008), 26–29.

[53] M. Stommel, K.-D. Kuhnert, Visual Alphabets on Different Levels of Abstraction for the Recognition of Deformable Objects. *Joint IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR), Cesme, Izmir, Turkey, August 18-20* **LNCS 6218** (2010), 213–222.

[54] E. Tanaka, Theoretical aspects of syntactic pattern recognition. *Pattern Recognition* **28** (1995) 7, 1053–1061.

[55] M. Varma, D. Ray, Learning The Discriminative Power-Invariance Trade-Off. *IEEE International Conference on Computer Vision (ICCV)* (2007).

[56] J. Woinowski, A Normal Form for Church-Rosser Language Systems. In: A. Middeldorp (ed.), *Rewriting Techniques and Applications*. Lecture Notes in Computer Science 2051, Springer Berlin / Heidelberg, 2001, 322–337. http://dx.doi.org/10.1007/3-540-45127-7_24

[57] L. Yang, R. Jin, R. Sukthankar, F. Jurie, Unifying Discriminative Visual Codebook Generation with Classifier Training for Object Category Recognition. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2008).

[58] E. Zhang, M. Mayo, Improving Bag-of-Words Model with Spatial Information. *Int'l Conf. on Image and Vision Computing New Zealand (IVCNZ)* (2010).