# 2D Context-Free Grammars: Mathematical Formulae Recognition

Daniel Průša and Václav Hlaváč

Czech Technical University, Faculty of Electrical Engineering
Department for Cybernetics, Center for Machine Perception
121 35 Prague 2, Karlovo náměstí 13
{prusa,hlavac}@cmp.felk.cvut.cz, http://cmp.felk.cvut.cz

**Abstract.** This contribution advocates that two-dimensional context-free grammars can be successfully used in the analysis of images containing objects that exhibit structural relations. The idea of structural construction is further developed. The approach can be made computationally efficient, practical and be able to cope with noise. We have developed and tested the method on a pilot study aiming at recognition of off-line mathematical formulae. The other novelty is not treating symbol segmentation in the image and structural analysis as two separate processes. This allows the system to recover from errors made in initial symbol segmentation.

## 1 Motivation and Taxonomy of Approaches

The paper serves two main purposes. First, it intends to point the reader's attention to the theory of two-dimensional (2D) languages. It focuses on context-free grammars having the potential to cope with structural relations in images. Second, the paper demonstrates on a pilot study concerning recognition of off-line hand written mathematical formulae that the 2D context-free grammars have the potential to deal with real-life noisy images.

The enthusiasm for grammar-based methods in pattern recognition from the 1970's [6] has gradually faded down due to inability to cope with errors and noise. Even mathematical linguistics, in which the formal grammar approach was pioneered [4], has tended to statistical methods since the 1990s.

M.I. Schlesinger from the Ukrainian Academy of Sciences in Kiev has been developing the 2D grammar-based pattern recognition theory in the context of engineering drawings analysis since the late 1970s. His theory was explicated in the 10th chapter of the monograph [17] in English for the first time.

The first author of this paper studied independently the theoretical limits of 2D grammars [14] and proved them to be rather restrictive.

The main motivation of the authors of the reported work is to discover to what extent the 2D grammars are applicable to practical image analysis.

This paper provides insight into an ongoing work on a pilot study aiming at off-line recognition of mathematical formulae. We have chosen this application domain because there is a clear structure in formulae and works of others exist which can be used for comparison.

Let us categorize the approaches to mathematical formulae recognition along two directions:

- *on-line* recognition (the timing of the pen strokes is available) versus *off-line* recognition (only an image is available).

– *printed* versus *hand-written* formulae.

We deal with off-line recognition of hand-written formulae in this contribution. Of course, the approach can be also applied to printed formulae.

## 2 State-of-the-Art

Formulae recognition has been a widely studied task. Several approaches from pattern recognition were adopted as well as new methods were motivated and designed by this particular task. A taxonomy of the methods is given in [3]. There are only a few commercial products performing formulae recognition. The most prominent is probably xMathJournal software [19] for Tablet PCs which uses on-line recognition. This software serves as a sophisticated calculator allowing inputs to be written by hand.

Most of the known methods follow the following two-phase procedure:

– Detection of individual symbols by image segmentation and labelling symbols using pattern recognition techniques.
– Structural analysis of relations among labelled symbols.

Classical approaches known from Optical Character Recognition were adopted to perform the symbols recognition phase. Images are segmented and a classifier is used to assign symbols to segments. There are also works performing symbol detection, segmentation and labelling during a single simultaneous process using Hidden Markov Models [18].

Formalisms related to structural analysis include geometric grammars, graph grammars, finding a minimal spanning tree, etc. [7, 12, 5].

Criticism of the commonly used methods concerns the image segmentation which is done without any knowledge of the formulae structure. It is hardly possible to recover from errors made during symbol segmentation phase. There have been attempts to employ additional error corrections schemes. However, this postprocessing corrections do not fit naturally into the pattern recognition process.

The approach based on the two-dimensional context-free (2D CF) grammars and a general structural construction tries to solve this problem [17]. The group from Kiev has applied their approach to images of musical scores [16] or electrical circuits [11]. Let us note that the notion of 2D CF grammars has appeared in works of other authors, e.g., [13].

## 3 Theory of Two-dimensional Languages

The theory of two-dimensional languages studies generalizations of formal languages to two dimensions. These generalizations can be done in many different ways. Automata working over a two-dimensional tape were firstly introduced by M. Blum and C. Hewitt, already in 1967. Since then, several formal models recognizing or generating two-dimensional objects have been proposed in the literature.

The most common two-dimensional object is a *picture* which is a matrix of symbols taken from a finite alphabet $\Sigma$. The set of all pictures over $\Sigma$ is denoted by $\Sigma^{**}$. Each subset $L \subseteq \Sigma^{**}$ is called a *two-dimensional language*. Note that it is also possible to consider objects of more general shapes, e.g. connected arrays, but we will work only with pictures.

One of the important tasks the theory of two-dimensional languages deals with is to search for suitable generalizations of the Chomsky hierarchy, especially of its first two levels, i.e. regular and context-free languages. Several formalisms were adopted or developed to fulfil this task, however, the more complex topology of pictures causes that the properties of the proposed classes usually differ to those known from regular, resp. context-free languages, despite the fact that the classes are defined via models resembling finite-state automata, resp. context-free grammars.

In the following sections, we describe two models of 2D finite-state devices and a 2D generalization of CF grammars. We also list basic properties of recognized (generated) languages.

### 3.1 Finite-state Devices

A *two-dimensional finite-state automaton* (2FSA) [15] is a natural generalization of the one-dimensional two-way automaton (which of recognition power equals the power of 1D finite-state automaton). It is equipped by a two-dimensional tape and allowed to move the head in four directions – left, right, up and down. This is the reason why it is also called a *four-way automaton* by some authors.

Let 2DFSA denote a deterministic 2FSA and $L(2FSA)$, resp. $L(2DFSA)$ the class of 2D languages recognizable by a 2FSA, resp. 2DFSA. The classes of recognized languages are characterized by the following facts:

- $L(2DFSA)$ is a proper subset of $L(2FSA)$.
- $L(2FSA)$ is not closed under concatenation (neither row or column), complement and projection.
- The emptiness problem is not decidable even for 2DFSA's.

Another kind of a finite-state device, so called *two-dimensional on-line tessellation automaton* (2OTA), was introduced by K. Inoue and A. Nakamura [9] in 1977. It is a kind of a bounded 2D cellular automaton. In comparison to the cellular automata, computations are performed in a restricted way – cells do not make transitions at every time-step, but rather a 'transition wave' passes once diagonally across them. Each cell changes its state depending on two neighbors – the top one and the left one. The result of a computation is determined by the state the bottom-right cell finishes in.

Again, let 2DOTA denote a deterministic 2OTA and $L(2OTA)$, resp. $L(2DOTA)$ be the classes of languages recognizable by the models. The most important results on the automata follow:

- $L(2DOTA)$ is a proper subset of $L(2OTA)$.
- $L(2OTA)$ is closed under row and column concatenation, union and intersection.
- $L(OTA)$ is not closed under complement while $L(DOTA)$ is closed under complement.
- $L(DOTA)$ and $L(FSA)$ are incomparable.
- $L(DFSA)$ is a proper subset of $L(DOTA)$.

D. Giammarresi and A. Restivo [8] present the class of languages recognizable by this device as the ground level class of the two-dimensional theory, prior to languages recognizable by two-dimensional finite-state automata. They argue that the proposed class fulfills more natural requirements on such a generalization. Moreover, it is possible to use several formalisms to define the class. Except tessellation automata, they

include tiling systems or monadic second order logic, thus the definition is robust as in the case of regular languages. On the other hand, 2OTA's are quite strong, since some NP-complete languages can be recognized by them. This result speaks against the promotion of the class to the 2D ground level class.

### 3.2 Two-dimensional Context-free Grammars

In this section we present a proposal of 2D CF grammars and results on them as they were given in [14]. The grammars are a generalization of 2D CF grammars introduced in [17].

Let $[a_{i,j}]_{m,n}$ denote the matrix

$$\begin{matrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{matrix}$$

For $P \in \Sigma^{**}$, let $\mathrm{rows}(P)$, resp. $\mathrm{cols}(P)$ denote the *number of rows*, resp. *columns* of $P$. We consider there is the only picture consisting of 0 rows and 0 columns. This picture, denoted $\Lambda$, is called the *empty picture*, analogously to the empty word $\lambda$.

We define two binary operations, the *row* and *column concatenation*. Let $A = [a_{i,j}]_{k,l}$ and $B = [b_{i,j}]_{m,n}$ be non-empty pictures over $\Sigma$. The column concatenation $A \oplus B$, resp. row concatenation $A \ominus B$ is defined if $k = m$, resp. $l = n$. The products of the operations are given by the following schemes:

$$A \oplus B = \begin{matrix} a_{1,1} & \dots & a_{1,l} & b_{1,1} & \dots & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{k,1} & \dots & a_{k,l} & b_{m,1} & \dots & b_{m,n} \end{matrix} \qquad A \ominus B = \begin{matrix} a_{1,1} & \dots & a_{1,l} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \dots & a_{k,l} \\ b_{1,1} & \dots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \dots & b_{m,n} \end{matrix}$$

It means $A \oplus B = [c_{i,j}]_{k,l+n}$, where

$$c_{i,j} = \begin{cases} a_{i,j} & \text{if } j \leq l \\ b_{i,j-l} & \text{otherwise} \end{cases}$$

and similarly, $A \ominus B = [d_{i,j}]_{k+m,l}$, where

$$d_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leq k \\ b_{i-k,j} & \text{otherwise} \end{cases}$$

The *generalized concatenation* is an unary operation $\bigoplus$ defined on a set of matrixes of elements that are pictures over some alphabet. For $i = 1, \dots, m$; $j = 1, \dots, n$, let $P_{i,j}$ be pictures over $\Sigma$. $\bigoplus [P_{i,j}]_{m,n}$ is defined if

$$\begin{gathered} \forall i \in \{1, \dots, m\} \ \mathrm{rows}(P_{i,1}) = \mathrm{rows}(P_{i,2}) = \dots = \mathrm{rows}(P_{i,n}) \\ \forall j \in \{1, \dots, n\} \ \mathrm{cols}(P_{1,j}) = \mathrm{cols}(P_{2,j}) = \dots = \mathrm{cols}(P_{m,j}) \end{gathered}$$

The result of the operation is $P_1 \ominus P_2 \ominus \dots \ominus P_m$, where each $P_k = P_{k,1} \oplus P_{k,2} \oplus \dots \oplus P_{k,n}$. See Figure 1 for an illustrative example.

| | | |
|---|---|---|
| $P_{1,1}$ | $P_{1,2}$ | $P_{1,3}$ |
| $P_{2,1}$ | $P_{2,2}$ | $P_{2,3}$ |

**Figure 1.** Scheme for the result of $\bigoplus[P_{i,j}]_{2,3}$ operation

**Definition 1.** A two-dimensional context-free grammar *(2CFG) is a tuple* $(V_N, V_T, S_0, \mathcal{P})$*, where*

- $V_N$ *is a finite set of nonterminals*
- $V_T$ *is a finite set of terminals*
- $S_0 \in V_N$ *is the initial nonterminal*
- $\mathcal{P}$ *is a finite set of productions of the form* $N \to W$*, where* $N \in V_N$ *and* $W \in (V_N \cup V_T)^{**} \setminus \{\Lambda\}$*. In addition,* $\mathcal{P}$ *may contain production* $S_0 \to \Lambda$*. In this case, no production in* $\mathcal{P}$ *contains* $S_0$ *as a part of its right-hand side.*

**Definition 2.** *Let* $G = (V_N, V_T, S_0, \mathcal{P})$ *be a 2CFG. We define a picture language* $L(G, N)$ *over* $V_T$ *for every* $N \in V_N$*. The definition is given by the following recurrent rules:*

*I)* *If* $N \to W$ *is a production in* $\mathcal{P}$ *and* $W \in {V_T}^{**}$*, then* $W$ *is in* $L(G, N)$*.*

*II)* *Let* $N \to [A_{i,j}]_{m,n}$ *be a production in* $\mathcal{P}$*, different to* $S_0 \to \Lambda$*, and* $P_{i,j}$ *(*$i = 1, \ldots, n$*;* $j = 1, \ldots, m$*) be pictures such that*
  - *if* $A_{i,j}$ *is a terminal, then* $P_{i,j} = A_{i,j}$
  - *if* $A_{i,j}$ *is a nonterminal, then* $P_{i,j} \in L(G, A_{i,j})$

  *Then, if* $\bigoplus[P_{i,j}]_{m,n}$ *is defined,* $\bigoplus[P_{i,j}]_{m,n}$ *is in* $L(G, N)$*.*

*The set* $L(G, N)$ *consists of pictures that can be obtained by applying a finite sequence of rules* $I$ *and* $II$*. The language* $L(G)$ *generated by the grammar* $G$ *is defined to be* $L(G) = L(G, S_0)$*.*

To illustrate the presented definition, let us show a simple example of a $2CFG$ that generates the set of all non-empty square pictures over $\Sigma = \{a\}$.

*Example 3.* Let $G = (V_N, V_T, S_0, \mathcal{P})$ be a $2CFG$, where $V_T = \{a\}$, $V_N = \{V, H, S_0\}$ and $\mathcal{P}$ consists of the following productions:

$$1)\quad H \to a\,, \qquad 2)\quad H \to a\,H\,, \qquad 3)\quad V \to a\,, \qquad 4)\quad V \to \begin{matrix} a \\ V \end{matrix}\,,$$

$$5)\quad S_0 \to a\,, \qquad 6)\quad S_0 \to \begin{matrix} a & H \\ V & S_0 \end{matrix}\,.$$

Productions 1), 2) are one-dimensional, thus it should be clear that $L(G, H)$ contains exactly all non-empty rows of $a$'s. And really, by applying rule $I$) on production 1), we have $a \in L(G, H)$. Furthermore, if $a^k \in L(G, H)$, then rule $II$) applied on production 2) gives $a^{k+1} \in L(G, H)$. Similarly, $L(G, V)$ contains non-empty columns of $a$'s.

By applying rule $I$) on production 5), the square $1 \times 1$ is generated by $G$. Since $a \in L(G, S_0) \cap L(G, H) \cap L(G, V)$, rule $II$) applied on production 6) gives that the square $2 \times 2$ is also in $L(G, S_0)$. The row, resp. column of length 2 is generated by

$H$, resp. $V$, thus rule $II$) can be applied again to produce the square $3 \times 3$, etc. By induction on the size, we can show that each non-empty square picture over $\{a\}$ can be generated and that there is no way to generate any non-square picture.

If we restrict right-hand sides of productions to be composed of at most two elements we obtain grammars from [17]. Let 2SCFG denote such a grammar and let us summarize the allowed types of productions:

P1. (column concatenation) $N \to A\,B$

P2. (row concatenation) $N \to \begin{matrix} A \\ B \end{matrix}$

P3. (renaming) $N \to A$

$N$ is a nonterminal, $A$ and $B$ are terminals or nonterminals.

A characterization of 2CFG's follows:

- $L(2CFG)$ is not comparable to $L(2FSA)$, neither to $L(2OTA)$.
- $L(2SCFG)$ is a proper subset of $L(2CFG)$.
- There is no analogy to the Chomsky normal form of productions.
- Time complexity of recognition depends on size of production's right-hand sides.
- The emptiness problem is not decidable.

We define two languages to demonstrate the incomparability between $L(2FSA)$ and $L(2CFG)$.

1. $L_1 = \{P \mid P \in \{a\}^{**} \wedge \mathrm{cols}(P) = \mathrm{rows}^2(P)\}$
2. Let $L_2$ be a language over $\{a, b\}$, consisting of square pictures, where each row and each column contains exactly one symbol $b$.

$L_1$ can be generated by a 2CFG, but it cannot be recognized by any 2FSA. On the other hand, $L_2$ is recognizable by a 2FSA (even by a 2DFSA), but it cannot be generated by any 2CFG.

The well known Cocke-Younger-Kasami algorithm [10, 20, 1] for recognition of languages generated by one-dimensional context-free grammars can be generalized on 2SCFG's [17]. Time complexity of the algorithm is

$$\mathcal{O}\left(m^2 n^2 (m+n)\right) ,$$

where $m$, resp. $n$ denote the number of rows, resp. columns of the picture. It is also possible to generalize the algorithm on languages generated by 2CFG's, but time complexity depends on sizes of production's right-hand sides in this case. Let $G = (V_T, V_N, S_0, \mathcal{P})$ be a 2CFG and

$$p = \mathbf{max}\left\{\mathrm{rows}(W) \mid N \to W \in \mathcal{P}\right\} , \qquad q = \mathbf{max}\left\{\mathrm{cols}(W) \mid N \to W \in \mathcal{P}\right\} .$$

Now, time complexity of the algorithm is

$$\mathcal{O}\left(m^{p+1} n^{q+1}\right) .$$

## 4 General Idea of Structural Construction

Principles related to the structural construction are explained in this section. The recognition process works with regions of the input image. A *region* is a connected set of pixels having some common property. The region is assigned a *label* determining which structure was recognized to be represented by the region (e.g., a fraction line as a part of a formula) and also a penalty giving the cost of derivation of the region. We consider two finite sets of labels: $V_T$ is a set of *terminals* and $V_N$ is a set of *nonterminals*. There is also one distinguished label $S_0 \in V_N$ corresponding to the whole structure we want to recognize.

At the beginning, there are so called terminal regions only, labelled by terminals. These regions can correspond to single pixels of the input image or to regions detected by an external tool. Usually, the property of regions is that they decompose an image into disjoint components. We relax this requirement and allow that regions to share some pixels.

A set of *rules* specifying how labelled regions can be combined to produce larger regions (their unions) is defined. A rule $N \rightarrow N_1, \ldots, N_k$ is interpreted as follows. If there are regions $R_1, \ldots, R_k$ labelled by $N_1, \ldots, N_k$, their sizes and positions fulfil some constraints connected to the rule then their union $R = \bigcup_{i=1}^{k} R_i$ with the label $N$ can be derived. The *penalty* of this derivation is computed from penalties of particular regions $R_i$. The rules are applied during an iterative process to derive larger and larger regions. The process ends when all possible regions have been derived. If the whole image was assigned by $S_0$ and the penalty of related derivation fits into some limit then the desired structure in the image was successfully recognized.

The described recognition process is too general and would be highly complex in time and space. Because of this, we need to seek some convenient unifications of rules and region shapes that will lead to an acceptable implementation. The formalisms resulting in these unifications can be based on 2D CF grammars we have already presented. Namely 2SCFG's are considered in [17]. In this case, permitted regions are only rectangles to suppress the complexity of derivations. Rules (or productions – to be consistent with the grammar terminology) are of types P1, P2 and P3.

Two observations can be made for the usage of the grammars: The complexity is still high and the constructs supported by the grammars are not rich enough to model relationships among symbols in mathematical formulae. We will address these two issues in the following section by introducing a suitable extension of the grammars.

## 5 Formulae Recognition Based on the Structural Construction

The main ideas taken from the structural construction we follow in our approach can be expressed in the following way:

Perform 'rough segmentation' of the input image. For each possible elementary symbol, find all occurrences of it and let the structural analysis decide, structure of which formula fits the input image best and how does the image segmentation looks like.

### 5.1 Specification and Goals of the Pilot Study

The pilot study was expected to support elements and constructs as numbers, variables, brackets, subscripts, superscripts, basic unary and binary operators, power to operations, fractions, sums, integrals and square roots.

We consider the inputs to be black and white images, however, the used method can be easily adopted to gray-scale images as well since it is general enough and does not depend on this assumption.

Our main goal was to investigate whether the structural construction can be successfully applied to off-line formulae recognition. In particular, we were interested how the method can deal with the following situations.

a) Symbols touching vertically or horizontally.
b) Symbols split into several components.
c) Ambiguities.
d) Misplaced symbols.
e) Noise.

Examples of these situations are given in Figure 2. Cases a) and b) demostrate standard formulae. Case c) illustrates a fraction line and a minus sign represented by the same image, the meaning is being given by the context. And finally, case d) includes an additional symbol $A$ that is by mistake placed into the formula. We require our method to exclude such a misplaced symbol and recognize the formula composed of the other symbols.

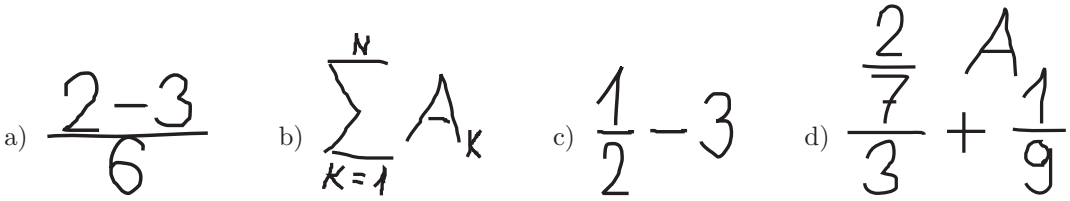


**Figure 2.** Corner cases we would like to handle

### 5.2 Extension of 2D Context-free Grammars

We use an extension of 2D CF grammars to model relationships among mathematical symbols. In this section, we give a description of the productions form and their application.

Let $N \rightarrow A \oplus B$ denote a production of our 2D CF grammar extension. The interpretation is similar to the interpretations of productions $N \rightarrow A|B$ and $N \rightarrow \frac{A}{B}$, regions labelled $A$ and $B$ can be united, producing a region labelled $N$. But this time we do not require the regions to touch each other. Permitted mutual positions of the regions are defined by a constraint that is connected to the *production*. The form of the constraint is depicted in Figure 3.

$R$ and $S$ are two regions in the image, $F$ is a *feature point* of $S$ (the center of left border in this particular case). $C$ is a dashed rectangle the size and position of which is given relative to the size and position of $R$. It represents the mentioned constraint. The considered production can be applied when the following conditions are fulfilled:

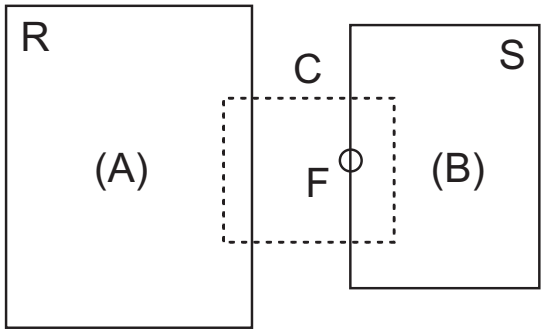


**Figure 3.** General production scheme of $N \rightarrow A \oplus B$

– $R$, resp. $S$ can be labelled by $A$, resp. $B$.
– feature point $F$ is located inside $C$.

The resulting region is the smallest rectangle containing $R$ and $S$. Feature points are some specific points of a region. We usually use corners and centers of bounding edges. We also compute baseline (when a new region is derived) and take the intersection with left or right bounding edge.

Figure 4 shows usage of a production to model 'power to' relationship where the constraint is defined for the bottom-left corner of the region storing the symbol four.

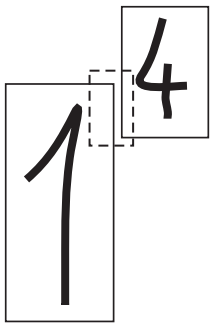

**Figure 4.** Example of the production usage

The penalty of the derived region is computed based on the following factors:

– Used production.
– Penalties of $R$ and $S$.
– Percentage of black pixels in the new region that are neither in $R$ nor $S$.
– Relative sizes and positions of $R$ and $S$.

To finish the description we will make three remarks. First, note that $R$ and $S$ can overlap. Second, we have defined the constraint on the location of $S$, assuming we know the location and size of $R$. In our structural analysis, we will also need the opposite case, i.e., to look for $S$ while knowing $R$. We slightly extend the production form by attaching one more constraint $C'$ of the mentioned meaning. And finally, in some cases of formulae constructs, it is more convenient to compose a new region from three regions instead of two (consider, e.g., a variable with a subscript and superscript). This can be easily supported by introducing productions of the from $N \rightarrow A_1 \oplus A_2 \oplus A_3$.

### 5.3 Architecture

Our software consists of two independent layers. The first layer performs *terminals detection*, while the second one is responsible for *structural analysis*. An external OCR tool is used within the first layer. Structural analysis is driven by a grammar defining supported mathematical formulae. The parsing algorithm is of a general nature and it can be used to recognize another types of structures if supported by a proper grammar.

### 5.4 Terminals Detection

We have developed an OCR tool for classification of image regions. The tool is based on a simple extraction of features from the input picture. The $k$-nearest neighbor classifier is implemented to classify the extracted vectors.

We have tested two methods for terminal symbols detection. The first method works with rectangular scanning windows of some specific sizes. Each of the windows is being moved trough the input image. Whenever it is in a new position, its content is evaluated by the OCR tool (provided that the number of black pixels exceeds some defined threshold). The result of the evaluation is a set of terminals assigned by a penalty, where the penalty corresponds to the belief that the scanning window stores a particular symbol. Only the terminals with a sufficiently low penalty are included in the result. For example, we have a window to detect subscripts and another window to detect variables and numbers. Sizes of the windows are determined by expected sizes of symbols in the formulae which means the method requires tuning for typical inputs.

It would be ideal in theory if we could use views of all sizes to scan the image, however, this approach is time expensive. Limiting the sizes of used views results in an acceptable performance but can miss some symbols in the image when their size differs from the expectation. Because of this we have tested the second method based on preprocessing the content of the image by computing connected components. Selection of views that are evaluated by the OCR tool is driven by these components. The bounding rectangles of the following areas are chosen:

- the components themselves,
- divisions of the components – up to two splitting horizontal and vertical points are considered,
- combinations of neighboring components.

### 5.5 Parsing Algorithm

We describe the algorithm that is used for the structural analysis phase. To simplify the description, we do not explain how information needed to track feature points and derivation trees is being updated. Just note that whenever a region $R$ is labelled by $N$, we record by which production this was derived.

1. Let $\mathcal{R}$ be a list of triples $(R, N, p)$, where $R$ is a region, $N$ label assigned to $R$ and $p$ penalty of this assignment. Initialize $\mathcal{R}$ by results of the terminals detection phase.
2. Iterate through $\mathcal{R}$. Let $(R, L, p) \in \mathcal{R}$ be the current element. For each production $N \to A \oplus B$ such that $L = A$ or $L = B$, take the rectangular area $C$ defined by the constraint of the production and find subset $\mathcal{S} \subseteq \mathcal{R}$, where for each $(R', L', p') \in \mathcal{S}$

the production can be applied on $R$ and $R'$. Let $\overline{R}$ be the derived region, $\overline{p}$ penalty of the derivation. If $\overline{p}$ is greater than some threshold then continue by the next iteration, otherwise check whether $\overline{R}$ has been already labelled by $N$. If not then append $(\overline{R}, N, \overline{p})$ at the end of $\mathcal{R}$. If there is already $(\overline{R}, N, p_2)$ in $\mathcal{R}$ and $\overline{p} < p_2$ then remove $(\overline{R}, N, p_2)$ from $\mathcal{R}$ and append $(\overline{R}, N, \overline{p})$, otherwise ignore the new derivation.

3. A formula is successfully recognized when the bounding rectangle of the input is labelled by $S_0$. If not then we can look for the largest region in $\mathcal{R}$ labelled by $S_0$.

To make the algorithm fast, we use a data structure storing points of a plane, allowing it to effectively evaluate queries of the type: for a rectangle, return the points that are located inside the rectangle (so called *orthogonal range searching*). A suitable data structure allowing to search in time $\mathcal{O}(\log n)$ can be constructed [2].

Our conclusions on time complexity are based on empirical data, we do not give an exact formula since it depends on many factors, including the number of the terminals detected during the first phase. Compared to the generalized Cocke-Younger-Kasami algorithm, time complexity is lower because the algorithm does not process all rectangles in the input. It would be possible to derive some upper bound on time, but it does not give a good idea about expected time. Instead of doing it, we rather discuss the most problematic case in the section regarding results.

## 6 Results

We have implemented the pilot study in Java. It includes an user interface allowing to browse formulae images, run the recognition on them and display results. Except the results, the interface also provides information helping to understand and tune the process of structural analysis. For example, it is possible to query for all regions labelled by a specific nonterminal, for penalties of related derivations, etc.

The implementation has been tested on over 200 handwritten formulae. We can conclude that after tuning the grammar there were no problems with correctness of the structural analysis. The problems we have encountered are connected mainly to the terminals detection phase.

We have faced some limitations when working with rectangular regions. Not all symbols in a formula can be separated by rectangles. Figure 5 a) shows one of the simplest examples. The bounding box of symbol $r$ contains a part of the subscript. It has an impact on recognition of $r$, which is assigned a bigger penalty in this case. In general, the recognition does not give good results, when the bounding boxes overlap too much. This is not usually case of printed formulae.

We were also forced to compose some elementary symbol from more components due to the mentioned limitation. A typical example is a square root which we consider to be formed of two parts (the square root argument can be treated as an additional, third part) – see Figure 5 b).

Other problems are connected to fraction lines. Continual subparts of a fraction line are also recognized as fraction lines. This leads to a large number of terminal symbols and possible combinations among them to be checked during the structural analysis. Figure 5 c) shows an image on which the problem starts to be visible. Recognition of this formulae takes about 20 seconds and grows fast for larger fractions (note that the recognition of each formula depicted in Figure 2 takes up to 2 seconds). We have implemented a preprocessing of detected fraction lines that reduces their

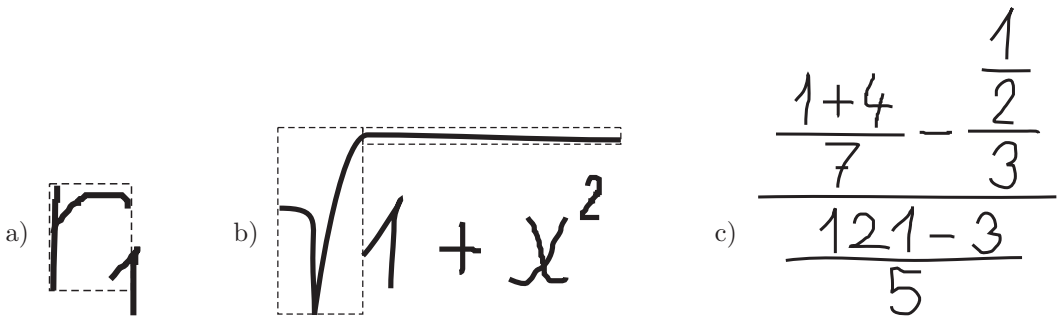


**Figure 5.** a) Bounding box of $r$ contains a part of the subscript. b) Square root symbol composed of two parts. c) Formula with several fraction lines.

number. It would be also possible to implement a special method (separated from the OCR tool) for fraction lines detection.

The last problem we have encountered was the accuracy of the used OCR tool that was not as high as it should be, but this problem can be solved by choosing a better tool.

## 7 Conclusions

We have showed that the method of structural construction can be applied for off-line mathematical formulae recognition. Our main contribution to the area of formulae recognition are the following achievements:

- Segmentation of the image is done during structural analysis (no error corrections are needed). We took advantage of the rich formula structure which allows this approach.
- Structural analysis is robust. It is penalty oriented and searches for the formula structure that best matches the input image. It can easily deal with noises, including misplaced symbols.
- We have designed an extension of 2D CF grammars powerful enough to express the formulae structure. It can be also effectively parsed (thanks to constraints defined via rectangles and the usage of data structures for orthogonal range searching).

We would like to focus more on printed formulae in the upcoming work. Our future plans include the usage of learning methods. Provided that a sufficiently large set of formulae is collected, the methods can be applied on learning etalons of terminal symbols and productions parameters. The learned etalons can improve the terminals detection phase, while the learned productions parameters will improve tuning of the grammar for a concrete typesetting style of formulae (so far we have tuned these parameters manually).

## References


[1] A. Aho and J. Ullman: *The theory of parsing, translation, and compiling*, vol. 1 – *Parsing*, Prentice-Hall, Englewood Cliff, New Jersey, 1971.

[2] M. Berg, O. Schwarzkopf, M. Kreveld, and M. Overmars: *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 2000.

[3] K.-F. Chan and D.-Y. Yeung: *Mathematical expression recognition: a survey.* IJDAR, 3(1) 2000, pp. 3–15.

[4] N. Chomsky: *Syntactic Structures*, Mouton and Co, The Hague, 1957.

[5] Y. Eto and M. Suzuki: *Mathematical formula recognition using virtual network*, in Proceedings of the ICDAR 2001, 2001, pp. 762–767.

[6] K. Fu: *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974.

[7] P. Garcia and B. Coüasnon: *Using a generic document recognition method for mathematical formulae recognition*, in GREC '01: Selected Papers from the Fourth International Workshop on Graphics Recognition Algorithms and Applications, D. Blostein and Y.-B. Kwon, eds., vol. 2390 of LNCS, Berlin, Germany, 2002, Springer-Verlag, pp. 236–244.

[8] D. Giammarresi and A. Restivo: *Recognizable picture languages*, in Int. J. of Pattern Recognition and Artificial Inteligence 6(2-3), 1992, pp. 32–45.

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

[10] T. Kasami: *An efficient recognition and syntax analysis algorithm for context-free languages*, Scientific report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, Mass., USA, 1965.

[11] V. Kiyko: *Recognition of objects in images of paper based line drawings*, in Third International Conference on Document Analysis and Recognition, Montreal, 1995, pp. 970–973.

[12] S. Lavirotte and L. Pottier: *Mathematical formula recognition using graph grammar*, in Proceedings of the SPIE 1998, vol. 3305, San Jose, CA, 1998, pp. 44–52.

[13] E. Miller and P. Viola: *Ambiguity and constraint in mathematical expression recognition*, in AAAI/IAAI, 1998, pp. 784–791.

[14] D. Průša: *Two-dimensional Languages*, PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2004.

[15] A. Rosenfeld: *Picture Languages - Formal Models of Picture Recognition*, Academic Press, New York, 1979.

[16] B. Savchynsky, M. Schlesinger, and M. Anochina: *Parsing and recognition of printed notes*, in Proceedings of the conference Control Systems and Computers, Kiev, Ukraine, 2003, pp. 30–38, in Russian, preprint in English available.

[17] M. Schlesinger and V. Hlaváč: *Ten lectures on statistical and structural pattern recognition*, vol. 24 of Computational Imaging and Vision, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[18] H. Winkler and M. Lang: *Online symbol segmentation and recognition in handwritten mathematical expressions*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, Munich, Germany, 1997, pp. 3377–3380.

[19] *Mathjournal 2.0*: a software for formulae recognition from the xThink company, 2006, http://www.xthink.com/MathJournal.html.

[20] D. Younger: *Recognition of context-free languages in time $n^3$*. Information and Control, 10 1967, pp. 189–208.