*Abstract*—This document presents a method of a logical links structure recognition between elements on diagrams. The applied approach intuitively mimics a human way of recognition that relies on merging already found connectors into more complex ones. This procedure is modeled by our method where simple and obvious connectors and gradually extended to more complex structures. Each iteration may lead to modification of connectors set obtained so far. The modifications are managed by a rules set describing logical and graphical constraints that should be satisfied by the connectors structure. If the extension leads to violation of constraints defined by the rules then the modification is not carried out. In this way, the recognized diagram structure is consistent with the assumed principles. The method was experimentally validated using the set of diagrams from three domains. In conclusions, method’s advantages and drawbacks are discussed.

1. INTRODUCTION

IN the last years there is a big interest in similar content image retrieval. There is plenty of research in this area. A deep survey is presented in [1]. The similar content image retrieval can be very helpful in automatic image annotation, in story illustration, copy detection, web image search and art image analysis. It can be also helpful in searching similar text documents if they contain images. Usually, images illustrate the content of document. They contain information included in document in condensed form and there is less problems with unambiguous comparison of expressed idea. This explains a strong interest in its application to similar patents search in order to speed up the procedure of patenting and to protect intellectual property rights. For automatic querying it is necessary to convert the information in the images into a high-level description. Usually, in the case of technical documents, images represent engineering drawings, diagrams, algorithms, operations and processes shown as charts. Images of this kind present the structure consisting of certain elements and connections between them. Automatic recognition of the connections structure is the first step towards further automatic analysis or even machine understanding of images. The problem is important and challenging, because the documents have highly-complex structure, tabular and graphical information is embedded and they contain conflicting technical jargon. Processing embedded images can aid to solve the problem. Typically, to apply such approach it is necessary to find images in the whole document. Then, images are processed in order to classify them to various classes: charts, diagrams, schemes, flowcharts, plots and photos. Next, a method dedicated to a given class of images is applied in order to recognize particular elements and their interconnections. It is worth noting that such image interpretation allows to write the content of an image in the electronic form, which facilitates its storage and further processing and comparison. In our research we focus on connectivity of elements in diagrams and flowcharts. The methods dedicated to this kind of graphics have to find: a) types of elements shown (various depicted shapes), b) segments of lines not belonging to found  
shapes, and c) connections created by these segments. Because diagrams usually contain texts embedded in diagram elements, it is also necessary to detect text areas, recognize it by applying OCR techniques and finally assign recognized texts to graphical diagram elements. The aim of the research described in this paper was to find a method that is able to retrieve logical links between elements depicted in the diagram. This logical structure is then  
expressed in an XML file. It is a difficult task especially when we consider that one connection may exist between more than two elements and the line segments constituting connectors can intersect.

The paper consists of six sections. The next section describes related works. Section III formulates problem to solve.The subsequent section presents the developed method. Section V experimentally validates our approach. Finally, some conclusions and recommendations related to further works are presented

1. RELATED WORKS

Early survey of works in this area is described in [2]. Theauthor writes that diagram recognition faces many challenges, including the great diversity in diagrammatic notations, and the presence of noise and ambiguity during the recognition process. Despite the flow of time from the year of this publication, all mentioned above features characterizing diagram interpretation constantly cause problems in chart recognition now

The paper [3] reviews research from the last decade. The authors present the whole process of chart recognition: chart segmentation, chart classification, chart interpretation and discuss existing solutions.

Relatively many works are devoted to online flowcharts recognition. In [4], the analysis to label each stroke of the flowchart and to group the strokes depending on the symbol they belong to is presented. The same area of research is represented in the paper [5]. In this paper the search for a suitable interpretation of the input is formulated as a combinatorial optimization task containing the max-sum problem. The recognition pipeline consists of two main stages. First, groups of strokes possibly representing symbols of a sketch (symbol candidates) are segmented and relations between them are detected. Second, a combination of symbol candidates best fitting the input is chosen by solving the optimization problem. The work [6] also concerns online charts but is focused on handdrawn electric circuit diagram recognition using 2D dynamic programming. The paper [7] presents another approach to hand drawn organizational diagrams that is based on Bayesian conditional random fields (BCRFs) that jointly analyzes all drawing elements in order to incorporate contextual cues. The classification of each object affects the classification of its neighbors. BCRFs allow flexible and correlated features. The online recognition of diagram is mainly applied in order to automatically check student tests.

Currently there is a great interest in flowchart recognition in the context of patent search. The paper [8] describes measures for assessing the effectiveness of flowchart recognition methods in the context of patent-related use cases. A survey of approaches can be found in [9]. A system for semi-automatic chart ground truth generation is introduced in the paper [10]. Using the system, the user is able to extract multiple levels of ground truth data.

Some works are devoted to chart recognition in documents. They apply various classification methods. In [11] spiking neural networks are used. The paper [12] presents a system for recognizing a large class of engineering drawings characterized by alternating instances of symbols and connection lines. The class of considered images includes domains such as: flowcharts, logic and electrical circuits, and chemical plant diagrams. The output of the system includes a list identifying the symbol types and interconnections. It may be used for design simulation or as a compact portable representation of the drawing. The method consists of two steps. First, domain independent rules are used to segment symbols from connection lines in the drawing image that has been thinned, vectorized, and preprocessed in routine ways. Then a drawing understanding subsystem works together with a set of domain specific matchers to classify symbols and correct errors automatically. They also proposed an interface to correct residual errors interactively.

Another important problem in diagram recognition and its automatic interpretation is recognition of texts appearing on diagrams. Separation of textual and graphical layers simplifies the further diagram structure analysis by reducing a number of involved graphical elements. It also makes it possible to attach textual information attributes to detected graphical elements of the diagram. In our approach we used the text separation method described in [13]. The method consists of three stages. In the first stage the text region candidates are elicited based on connected components analysis and some simple geometrical properties of connected components clusters. At the second stage, pattern recognition methods are applied to the set of candidates to discriminate between true text areas and other "false" candidates. Finally, OCR is applied to candidate regions and the final text region set is refined based on the analysis of the contents of the OCR-recognized strings.

1. PROBLEM FORMULATION

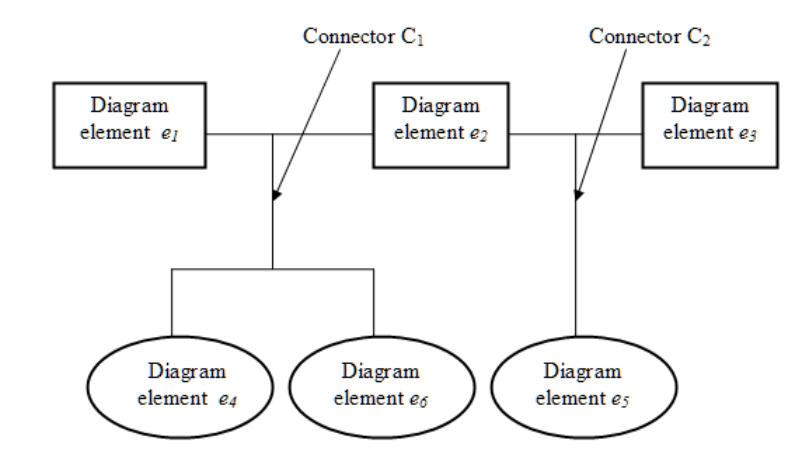
In the further part of this article by a *diagram* we will mean a drawing that shows a set of entities - *diagram elements* (DEs) and connections between them. In the literature, the term "diagram" is used interchangeably with "chart", but diagram seems to be more general. We will be considering diagram images created with appropriate software or precisely drawn manually using drawing tools (rulers, drafting templates) and then converted into raster images by scanning. Hand-sketched diagrams are out of the scope of this article due to big inaccuracies appearing in this type of drawings. Our aim is to retrieve the logical structure of the diagram, so that it corresponds to intents of the diagram author. Informally, by the logical structure of the diagram we mean here the links between diagram elements. Diagram elements represent various items appearing in the real world modeled by the diagram. Their meaning depends on the domain of application. In the case of program flowcharts they can represent: statements, code blocks, conditions, data sources etc. In a logical circuit diagram they represent gates and functional blocks like: registers, multiplexers, flip-flops, etc. In organizational charts their elements are usually officials or departments. Although the method described here can be applied to any kind of diagrams, we mainly focus on organizational charts and program flowcharts. The element of the diagram is depicted by a simple 2D geometric shape like: rectangle, circle ellipse, rhombus, diamond. There are some attributes assigned to diagram elements. The basic attributes of an element are the kind of 2D shape and the textual description (the text usually inscribed into the element shape). Additional attributes that can be easily retrieved from the diagram image are: the shape interior color, shape line color and the shape border line width. They can be meaningful in certain types of diagrams, in other types of diagrams they may be ignored. The methods used to recognize shapes appearing in diagrams and to evaluate their attributes will be shortly described in the section IV-I.

The link in the diagram represents a logical relation or an association between DEs. Links are graphically represented by polylines or sets of intersecting or connected polylines, which endpoints are in the close vicinity of DEs being connected. Depending on an application domain, various types of links can be distinguished. The simplest links are one-to-one links which represent the association between a pair of DEs. The many-to-many link is the more complex case that associates the larger set of DEs. The links can be undirected or directed. In directed links some polyline endpoints are arrows. The directed link usually indicates the information flow or organizational subordination. If directed links are used the many-to many association may turn into one-to-many association where only the single endpoint of the link is not arrowed, while all remaining endpoints are arrows. The graphical representation of the link will be further called *connector*. The connector is therefore the set of polylines that intersect each other or constitute T-style junctions. Finding connectors that are known to represent one-to-one links is a simple technical problem. Also in the case where it is known that the connection of polylines belonging to a common connector is indicated by dots (or other graphical marks) placed at connection points, the problem only lies in reliable connection marks recognition. It is however much more complicated if we cannot assume that connections of polylines belonging to a connector are graphically marked by connection marks. In such case the recognition of diagram structure must be based on the trial to "guess" the diagram author intention. In the further part of the paper, we will describe the method that deals with this kind of diagrams.

We are considering here the method which starts with vectorized diagram image on its input. The vector representation of the original (raster) image is obtained by applying the sequence of image processing operations followed by the vectorization procedure that converts a binary image into the set of line segments (vectors). Let us also assume that DEs have been already successfully found. In our approach we used the shapes recognition method based on vector sequence matching to basic shapes defined either by rules or algebraically, as described shortly in section IV-I. The vectors constituting found DEs were identified and extracted from further considerations. Let E = {e1, e2, ..., eN } denote the set of found DEs and let L denote the set of line segments (called later *edges*) not assigned to any detected DE. The edge is a pair of 2D points being its ends on the plane lj = (p0j, p1j), pi = (xi, yi). Line segments in L possibly belong to connectors. Formally, a connector is a subset of connected edges from L. Our aim at this stage is to gather as many as possible edges into disjoint subsets Ci ⊆ L corresponding to connectors, while leaving as little as possible lines unassigned, i.e. belonging to the unassigned set U. The result is the family of subsets {C1, C2, ..., CM }. The construction of connector sets can be considered as a rulebased process, where rules define some constraints that must be satisfied in order to put a certain subset of edges into a connector, as well as principles that force inclusion of some edges in a single connector. Rules determining principles of reasonable construction of connectors were derived from the analysis of diagram structures appearing on typical diagram images. The analysis has been carried out using the set of diagram images from various domains, that we used for testing of our diagram analysis methods. The rules defining the construction of a connector Ci of L are as follows:

• the elements of Ci are coherent (i.e. for each pair of edges in Ci there is a sequence of other edges in Ci, possibly empty, that connects them);  
• if there is an edge in Ci that has a vertex not shared with other lines in Ci (i.e. it is the endpoint of a polyline which elements are within Ci) then it must be close to one of detected DEs from the set E, such a vertex is a *terminal vertex*;  
• there are no two vertices of edges in Ci that are terminal vertices and are close to the same DE (the connector consisting just of a single edge cannot link the DE with itself);  
• there are no cycles in the graph defined by the set Ci, i.e. there are no polylines in Ci that intersect with themselves;  
• if there is an edge la in Ci that has a common vertex with another edge lb in L and lb is not assigned to any other connector Cj then lb must also belong to Ci (no connectors ending in the middle of polylines);  
• if p is an internal vertex shared by two edges belonging to Ci then it cannot be closer than ǫ to an internal vertex shared by two edges belonging to another connector Cj (connectors cannot touch each other, except of the case  
that the terminal edge endpoint touches a terminal edge endpoint of another connector);  
• each edge is uniquely assigned to one of subsets Ci or to U (edges are not shared by connectors);  
• the longest path connecting two terminal vertices in Ci cannot be longer than assumed threshold i.e. 8 (no connectors of very complicated shape);  
• the angle between two adjacent edges in Ci which do not share the common vertex with any other edge within the same connector is not smaller than the right angle (no acute angles in the polyline segments of connectors);  
• width of each edge within Ci does not differ by more than 30% from the weighted average line width in the connector;  
• two elements ei, ej ∈ E cannot be connected along more than one path within the single connector (they can be however connected by many paths, provided that they belong to different connectors).  
 Because it seems reasonable to merge connectors into more complex ones, as far as it does not lead to violation of rules presented above, then the ultimate aim is to partition the set L into the family of subsets {U, Ci : i = 1, ..., M)} so as to minimize the number of subsets Ci with the additional constraint that the set U does not contain any subset of edges that constitutes a valid connector.

For practical purposes related to automatic analysis of the diagrams it is essential not just to find connectors (being the sets of edges) but rather to determine the sets of connected Des which are connected by individual connectors. Therefore, the final result of the structure recognition procedure is the family of sets of DEs, where each set defines elements connected by the single connector. Additionally, because each connector endpoint can be marked with an arrow, this information should be also retrieved and included in the data structure being the output of the recognition procedure. Each endpoint of the connector is described by a pair (e, t) where e ∈ E and t ∈ {true, false} indicates whether or not the endpoint of the connector linked to e is an arrow. The elements connected by the single connector C can be specified by the multiset VC = {(ei, ti) : ei ∈ E, ti ∈ {true, false}}. The expression t = true denotes the arrow appearance and t = false. denotes arrow-less connector endpoint. Multisets are applied here instead of simple sets because the diagram element can be connected with itself. In such case there exist two connector endpoints associated with the same element. It leads to the appearance of the pair containing this element twice in VC. Finally, the product of the diagram structure recognition is the set of connectors and the family of corresponding multisets of connected diagram elements



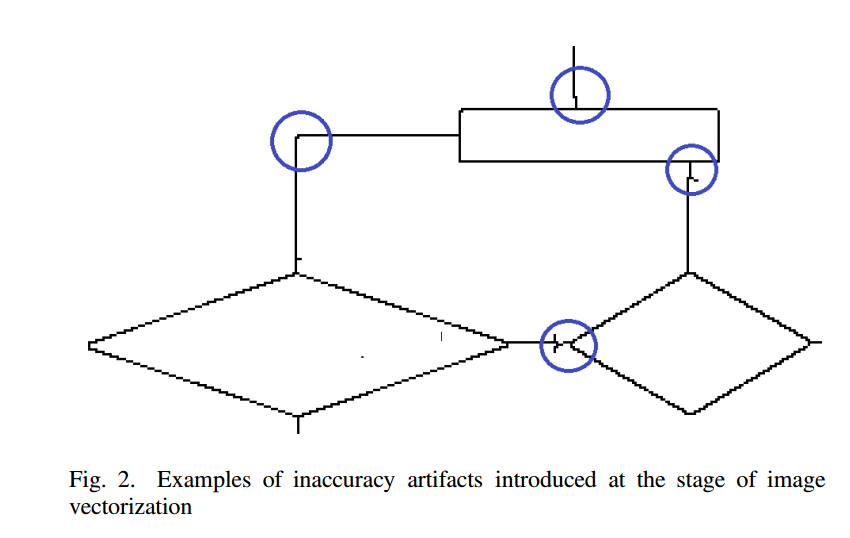
(C, b Ub) = ({Ci : i = 1, ..., M}, {VCi : i = 1, ..., M}), (1)

where M is the number of detected connectors. A DE may belong to more than one multiset VCi if it is connected with other DEs by various connectors. The case where the central element is shared by two connectors is shown in Fig.1. The elements of the diagram are connected by two connectors: C1 and C2. In this case, the structure recognition procedure builds the following diagram description: (C, b Ub) = ({C1, C2}, {{(e1, false), (e2, false), (e4, false), (e6, false)}, (2) {(e2, false), (e3, false), (e5, false)}}).  
IV. DESCRIPTION OF THE METHOD  
 Unfortunately, the formulation of the diagram structure recognition problem does not lead to an efficient solution, other that "brute force" approach based on exhaustive search of all subsets of L, which is obviously infeasible in most practical cases. Therefore, we propose simplified suboptimal solution that leads to construction of a connector set {C1, C2, ..., CM }, which however does not guarantee that the minimal number of connectors are found. On the other hand however, it applies some intuitive principles that humans typically apply when trying to read a structure on a diagram presented on an image. Experiments described in section V show that diagram structure recognized with the proposed algorithm is close to the human interpretation of test diagram images. The approach taken consists in the observation that when a human tries to find connectors in a diagram by sight, it intuitively starts with long edges and tries to interpret them as "simple connectors" connecting pairs of DEs. Then a human tries to find "branches" that connect other DEs to previously found simple connectors. Next, one tries to find inter-connectors, i.e. polylines that connect previously found connectors. Finally, we (humans) try to merge already found connectors into more complex ones by finding intersecting lines belonging to various connectors that are candidates for merging. It is a process that starts with simple and obvious connectors and gradually extend them to more complex structures. This process can be modeled as a procedure implemented in a computer. Each stage outlined above is in fact an iterative operation that processes successive items (edges, polylines, simpler connectors - depending on the stage), where each iteration may lead to modification of the connector set obtained so far. The modification is however conditioned on the rules set presented in the previous section. If it leads to violation of constraints defined by the rules then the modification is not carried out. In this way, at each stage of the procedure we have the diagram structure that is consistent with the assumed principles

Now the procedure will be described in more details. The input to the procedure is the set of diagram elements E and the set of line segments (vectors) obtained from the raster image vectorization procedure (vectorizer). The applied vectorization procedure is based on the algorithm described in [14]. We will not deal here with the details of methods of shape recognition used to obtain the set E. They are briefly described in the subsection IV-I. In the proposed algorithm, the set of constants usually applied as thresholds are utilized. Values of these thresholds were estimated experimentally by analyzing a set of typical diagram images from the validation set. The selected validation set is disjoint from the testing set used in order to evaluate the method performance. The connectors finding method consists of the following steps:

*A. Edge detection*

The aim of this step is to create the set of edges L from the set of line segments fetched by the vectorizer. We use the term "edge" to emphasize the difference in relation to the notion of simple line segment which is the direct product of vectorization. The line segments created by the vectorizer should not be used directly in further steps of the procedure. Our experiments showed that there are some troublesome artifacts, especially at intersections of relatively thick lines or at vertices of polylines. They are short line segments of the length of single pixels that connect longer line segments. Such geometrical structures need a kind of smoothing in order to obtain longer and straight line segments, most likely being "true" lines in the original diagram. Here we call such smoothed and merged line segments "edges". An example of an erroneous line segment structure created by the vectorizer is presented in Fig.2. The unwanted artifacts are indicated by blue circles.



The procedure takes the line segment that is not yet assigned to any element in E nor to any edge in L and tries to extend it to a longer edge by building a polyline being a sequence of interconnected line segments. In each iteration the algorithm tries to attach the next line segment that is adjacent to one of end points of the already created polyline. This segment is selected that is most collinear with the straight line approximating already created polyline. The attachment criterion is used to select candidates for extension. It takes into account the angle between the candidate segment and already approximated line and the length of the candidate. Short segments (being probably artifacts of the vectorization procedure) can be connected even if the angle is relatively big. The logical predicate used as the extension criterion is as follows:

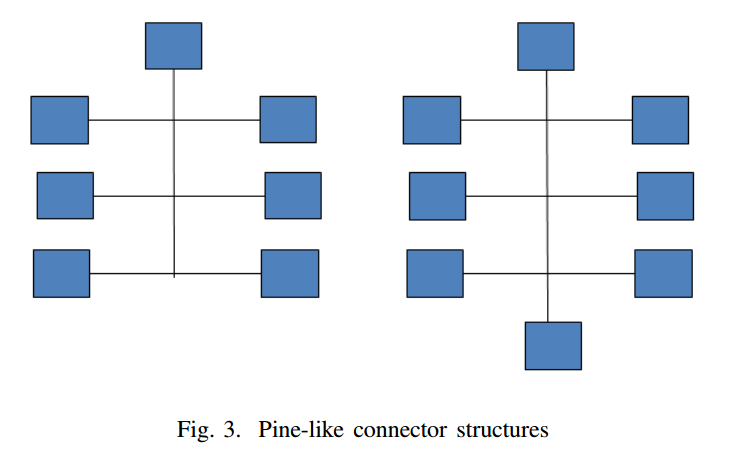
∠(e, c) < αmax ∨ len(lt) ≤ lmin ∨ len(c) ≤ 1.5 ∗ w, (3)

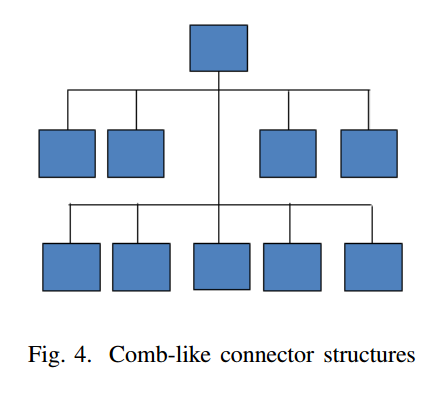
where e is the edge (single line segment) approximating the polyline created so far, c is the line segment - the candidate for extension, lt is the terminal line segment in the polyline that is adjacent to c and len(•) is the length of the line segment. w is the average width of the line segments already attached to the polyline. αmax was experimentally set to 10◦.

According to the criterion (3), the angle between connected segments must be small enough or at least one of adjacent segments (the candidate one or the terminal segment) is short enough. This alternative makes it possible to use residual vectors of the length of 1-2 pixels that are artifacts of the vectorization procedure. Such residual vectors often are oriented at big angles with the relation to its (longer) neighbors. The procedure iteratively tries to extend the polyline until no new segments can be attached. After each extension the new linear approximation of the polyline (denoted by e in the formula 3) is evaluated by least square fitting of points on the polyline to the approximating line.

*B. Preparation to processing*The aim of this stage is to identify DEs that are close enough to endpoints of edges created in the previous stage. In this way it is possible to identify edges that are candidates for terminal edges of connectors. The terminal edge of the connector is the edge directly attached to the element connected by a connector. Then for all edge vertices the closest DE is found that is within assumed maximal allowed distance from the vertex. The tolerance is estimated depending of the shape and edge line widths. By analyzing the set of exemplary diagrams we assumed that the tolerance should be evaluated as min(3 ∗ max(we, ws), 0.25 ∗ sBB) where we is the width of the edge line, ws is the width of the line of the DE shape and sBB is the smaller of x and y sizes of the bounding box enclosing the element. The angle between the closest element edge and the connector edge is also taken into account to avoid considering as very close an edge that is almost parallel to an edge of DE. As the result of this stage, each edge endpoint is annotated either with the index of the close diagram element or with "dummy" index, denoting that there is no close element to the edge endpoint. Additionally, the "edge structure" is created, that makes it possible to quickly find all edges in L adjacent to a given vertex.

*C. Finding simple connectors*At this stage simple connectors are being found, where two DEs are directly connected by a single edge. It lies in finding edges with two endpoints marked with various diagram elements. Cases where the edge connects the shape with itself and is completely within this shape bounding box are excluded by applying one of constraints defined in Section III.  
*D. Finding polyline connectors*This stage consists in finding edge sequences (a polyline consisting of edges) that connect two shapes. The procedure used here starts with a polyline consisting of a single edge that is not yet assigned to any connector. It iteratively tries to extend the polyline by attaching its left/right endpoint neighbors until no further extension is possible or the attached edge is connected to an element. Backtracking is applied in cases where there are many neighboring edges adjacent to the terminal edge in the polyline and extending edge selection in certain iteration leads to the polyline that neither can be further extended nor it terminates with the edge adjacent to any DE.  
 The procedure is being repeated, each time starting with an edge that is not yet assigned to any connector. If it leads to the polyline connecting two diagram elements then it is assumed to be the polyline connector. The new connector is then created and all edges are labeled as assigned to this connector.  
*E. Finding inter-connectors* In the previous stages only such connectors (or fragments of connectors) were recognized which link pairs of DEs. In the next phase, new polyline connectors are tried that connect already found connectors with other connectors or diagram elements. The interconnector appears for instance in the diagram in Fig.1. At the first stage, the simple connector between elements e1 and e2 is found. The polyline connector linking elements e4 and e6 is detected in the second stage. In the current stage the interconnector that links this two simpler connectors will be recognized. The case of "branch" that extends simpler connectors appears in Fig.1 in the case of the single connector that links elements e2 and e3. The branch consisting of the vertical edge in the right part of the diagram links the simple connector with the element e5.





The procedure iterates until a single iteration does not result in any extension of the obtained connector sets. The single iteration in turn, consists of subiterations that iterate over all unassigned line segments, where each unassigned segment is tried to be extended into a polyline connecting two earlier detected connectors or a connector and a diagram element. Typically, this stage creates T-connections. T-connection is the connection of edges, where one edge perpendicular to another one touches them in the middle. The connection between the edge linking e2 with e3 and the vertical edge adjacent to e5 is a typical T-connection.

The next two stages are aimed on merging connectors found earlier into more complex ones. Pairs of connectors that are candidates to merging must have intersecting edges. Merging all connectors having intersecting edges in many cases would lead to a structure not intended by the diagram author. Actually, the problem of simpler connector merging seems to be the hardest one in the process of diagram structure recognition. We distinguished two specific graphical configurations of edges that are typically used when composing diagrams and we perform connector merging only if the merged connector conforms to the one of these specific configurations. We called these configurations *pine-like* and *comb-like* structures. The structures are shown in Fig.3 and Fig.4.

*F. Constructing pine-like connectors*In this step, the specific type of connector is detected which consists of a simple single edge that intersects other connectors. It is assumed that in such case the diagram creator  
intention was to depict the situation where elements connected by such a connector are connected each to another. In order to find pine-like connectors, the simple connectors  
(being just single edges) are tested against intersection with edges of other connectors, let us call it *trunk*. All other simple connectors that intersect the trunk are merged to the group containing the trunk.

*G. Constructing comb-like connectors*The comb-like structure is presented in Fig.4. It consists of the vertical trunk connector intersected by one or more "combs". Comb is a connector with the principal horizontal edge to which a series of simple vertical edges (branches) are T-connected. Additionally, it is required that comb teeth are approximately of to the same length in the interval < 0.1 ∗ l pr, 0.7 ∗ lpr > where lpr is the length of the principal comb edge. The comb-like structure must also have trunk – the vertical line being a connector that have a single DE at its top and the trunk length must be at least 2 ∗ ltmin, where ltmin is the shortest tooth length of the comb. The procedure of comblike connector construction consists in finding connectors that satisfy aforementioned conditions. The set of connectors that satisfies it is replaced in the set {C1, C@, ..., CM } by the product of the merge operation.

*H. Merging connectors by dot-markers*Finally, all these pairs of connectors are merged that contain lines that cross one with another, where there is a dot-marker at the intersection of lines belonging to various connectors. In order to consider two line segments as marked for merging, the following conditions must be satisfied: • intersecting lines are approximately of the same width  
(0.5 ≤ w1/w2 ≤ 2.0), where w1, w2 are widths of lines; • lines are approximately of the same color, the color tolerance is defined for the components of 24-bit RGB color space; Rmax - Rmin < 30,

|  |  |
| --- | --- |
| Gmax - Gmin < 30, | (4) |
| Bmax - Bmin < 30, |  |

• the angle between intersecting lines is approximately the right angle (with the tolerance range from 80◦ to 100◦);  
• the four diagonal pixels at the distance p(2) ∗ max(w1, w2) from the lines intersection point are closer in colors to the average lines color than to the background color. The last condition is responsible for detecting the dot intersection marker at the lines intersection point.

*I. Diagram elements recognition*Because this paper is mainly focused on the recognition of connections between diagram elements, we will only briefly describe methods used in order to recognize diagram elements. We assume that diagram elements are: a) polygons, b) circles, ellipses or arches of ellipses and c) shapes being combination of a) and b), e.g. the symbol of a drum often used in flowcharts to denote mass storage. Methods used to recognize polygonal shapes are based on rules that define the mutual geometrical relations between line segments constituting polygon edges. For example, the parallelogram not being just a rectangle is defined as the sequence of 4 edges (l0, l1, l2, l3) defined by their endpoints (p(iB), e(iE)), i = 0, ..., 3) that satisfy the set of constraints:

• | p(iE) - p( i⊕B1) |≤ ǫ for i = 0, ..., 3;  
• either p(iE) = p(i⊕B1) or p( iE) is connected with p( i⊕B1) by a chain of short line segments that are entirely included inside the bounding box defined by p( iE) and p( i⊕B1);  
• l0 k l2 and l1 k l3;  
• ∠(l0, l1) ≤ 90◦ - αtoll or ∠(l0, l1) ≥ 90◦ + αtoll,

where i ⊕ 1 = (i + 1) mod 4 and αtoll denotes the tolerance for right angles. The last constraint makes it possible to distinguish between rectangles and other parallelograms.

The procedure of shape recognition starts with a line segment from the vector set created by the vectorization procedure and successively tries to extend it to a sequence of segments, so that the constraints defined for allowed shapes are satisfied. It may happen that certain sequence of line segments satisfies constraints for more that single shape. For example, due to drawing inaccuracies, a quadrilateral found in the diagram may satisfy both constraints for the rectangle, the trapezoid as well as for the rounded vertices rectangle. In such a case, the measure of inaccuracy for all candidate shapes is computed and this shape is finally selected for which the inaccuracy measure is lowest. The procedure is repeated, each time starting with a next line segments that is not already assigned to any shape, until all unassigned line segments are tried.

In the case of ellipses and arches the procedure starts with a candidate edge and tries to extend it to a polyline that best approximates the ellipse fragment. Only axis-aligned ellipses are considered. Let S = (l1, l2, ..., ln) be a sequence of line segments that approximate an ellipse arch. At each stage the procedure tries to extend it with one of line segments that is adjacent to l1 or ln. Such extending segment is selected for which the ellipse approximation error is the lowest. The approximation error is the average distance of pixels constituting the polyline S to the best fitting ellipse. The pixel set used for approximation is created by applying Bresenham line drawing algorithm to all lines in the set S. The axis-aligned ellipse is defined by four parameters: coordinates of the ellipse center (xc, yc), the length of x-axis dx and the shape factor a - the ratio of x and y axes lengths a = dx/dy. The best fitting ellipse is found using the method described in [15]. The parameters of the optimal ellipse as assumed to be within the reasonable ranges determined in relation to the image size, e.g. the ellipse (xc, yc) center must be within the range defined by the image resolution and both ellipse axes must be not longer than the corresponding image size along x or y axes. If the parameters computed by the optimization procedure are out of these ranges then the approximation is assumed to fail and another extending line is tried. The extension is continued until the closed ellipse is obtained or no more lines can be attached. The final ellipse arch is accepted if it constitutes at least 50% of the complete ellipse. This acceptance threshold may seem to be high, but shapes that we consider here as diagram elements never consist of shorter ellipse fragments. On the other hand, setting too low value of the threshold may lead to false recognition of other shape elements as ellipse fragments.

*A. Evaluation of diagram structure recognition accuracy*The accuracy of diagram structure recognition can be assessed by the complexity of operations necessary to convert the recognized structure into the correct one (ground truth). This complexity can be measured by the summed cost of elementary operations that can be used in order to turn the recognized structure into the correct one. We focus here on the evaluation of the multiset Ub as defined in (1). Let us assume that the recognized structure described by Ub is to be converted into the correct structure Ub ∗ by applying the sequence of elementary operations. In the result, the sequence of structures is created: (Ub = Ub1, Ub2, ..., UbK = Ub ∗) where Ubk is converted into Ubk+1 by applying one of the following elementary operations from the set O = {oC, oS, oM, oC, oE, oR, oA, oD} defined as follows: