Image Synthesis from Natural Language Description

Nicolae Ţăndăreanu[1] and Mihaela Ghindeanu[2]
Faculty of Mathematics and Computer Science,
Department of Computer Science,
University of Craiova, Romania

[1] ntand@oltenia.ro
[2] mghindeanu@yahoo.com

Abstract. In this paper we present another application of the labelled stratified graphs introduced in [2]. More exactly, we consider a text T given in a natural language. T gives a description of some world of objects. Using a recursive transition network ([1]) the information contained in T is extracted. As a consequence, a labelled graph G is obtained. The structure of G is extended to a labelled stratified graph and applying the appropriate methods we can realize the synthesis of the image specified in T. The method is implemented and all the formal computations are presented. In conclusion, taking into account a text describing an image, we present an application of the labelled stratified graphs in image synthesis.
Keywords: labelled stratified graph, knowledge representation, natural language, recursive transition network
AMS Subject Classification: 68T30, 68T45, 68T50

1 General presentation

The concept of *labelled stratified graph* (LSG) was introduced in [2] in connection with that of knowledge base with output (KBO). Various applications of this concept in travel scheduling, attribute graphs and transport problems where presented. In general, this concept is used in connection with knowledge representation. As a general presentation of this method we can say that in the first step a labelled graph G is obtained by extracting the information from a given knowledge piece. In the second step a LSG is built over G. In the last step, the algebraic mechanism offered by a LSG is used to formalize the reasoning process.

In this paper the following aspects are treated:

1) We give an algorithm by means of which we are able to extract the information from the text of a knowledge piece KP. The information is represented by objects and the binary relations between them. Automatically, the drawing of the corresponding labelled graph G is realized. In order to process the information from KP a *recursive transition network* is used. A

LSG over G is defined and based on this structure an appropriate reasoning environment is obtained.
2) The above method is exemplified and the means offered by a LSG are used to reconstruct the image of the world described in KP. We used here the term "reconstruct" for the following reason: the method can be applied in a communication from the location A to the remote location B in order to reconstruct in B the image observed in A. In this case a text description of the scene is transmitted instead of an image. The text is transmitted from A and based on this description an appropriate image is reconstructed in B.

The paper is organized as follows: in Section 2 a concise presentation of the concept of labelled stratified graph is given; in Section 3 we give a general algorithm to draw automatically the labelled graph associated to a text description given in a natural language whose syntax is represented by means of a recursive transition network; based on the concept of LSG we present in Section 4 the manner in which we can synthetize the image described by a text description; Section 5 includes an implementation of the method presented in this paper; in the last section several future problems are presented.

2 Labelled Stratified Graphs

In this section we give a brief presentation of the concept of *labelled stratified graph* (shortly, LSG), which is a basic one for the next sections.

Intuitively, a labelled graph is a directed graph such that each arc contains an entity called label such that, finally, this entity carries a specific information.

Definition 1. *A labelled graph is a tuple* $G = (S, L_0, T_0, f_0)$*, where*

- S *is a finite set of elements called nodes*
- L_0 *is a finite set of elements named labels*
- T_0 *is a set of binary relations on* S
- $f_0 : L_0 \longrightarrow T_0$ *is a surjective function*

Such a structure admits a graphical representation. Each element of S is represented by a rectangle specifying the corresponding node. We draw an arc from $x \in S$ to $y \in S$ and this arc is labelled by $a \in L_0$ if $(x, y) \in f_0(a)$. This representation is shown in figure 1.

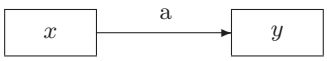

Fig. 1. A labelled arc

We consider a labelled graph $G = (S, L_0, T_0, f_0)$ and a symbol σ of arity 2. We define recursively

$$\begin{cases} B_0 = L_0 \\ \\ B_{n+1} = B_n \cup \{\sigma(x_1, x_2) \mid (x_1, x_2) \in B_n \times B_n\}, n \geq 0 \end{cases} \tag{1}$$

The set $B = \bigcup_{n\geq 0} B_n$ gives the support set of a *Peano algebra* generated by L_0 ([**?**]).

By $Initial(L_0)$ we denote some collection of subsets of B. Namely, we say that $M \in Initial(L_0)$ if the following conditions are fulfilled:

- $L_0 \subseteq M \subseteq B$
- if $\sigma(u, v) \in M$ then $u \in M$ and $v \in M$

We define the mapping $prod_S : dom(prod_S) \longrightarrow 2^{S\times S}$ as follows:

$dom(prod_S) = \{(\rho_1, \rho_2) \in 2^{S\times S} \times 2^{S\times S} \mid \rho_1 \circ \rho_2 \neq \emptyset\}$
$prod_S(\rho_1, \rho_2) = \rho_1 \circ \rho_2$

where $\rho_1 \circ \rho_2$ represents the product of the binary relation ρ_1 and ρ_2:

$$\rho_1 \circ \rho_2 = \{(x, y) \mid \exists z : (x, z) \in \rho_1, (z, y) \in \rho_2\}$$

We denote by $R(prod_S)$ the set of all the restrictions of the mapping $prod_S$:

$$R(prod_S) = \{u \mid u \prec prod_S\}$$

Let u be an element of $R(prod_S)$. We denote by $Cl_u(T_0)$ the *closure* of T_0 in the partial algebra $(2^{S\times S}, \{u\})$. This is the smallest subset Q of $2^{S\times S}$ such that $T_0 \subseteq Q$ and Q is closed under u. It is known that this is the union $\bigcup_{n\geq 0} X_n$, where

$$\begin{cases} X_0 = T_0 \\ \\ X_{n+1} = X_n \cup \{u(\rho_1, \rho_2) \mid (\rho_1, \rho_2) \in dom(u) \cap (X_n \times X_n)\}, n \geq 0 \end{cases} \tag{2}$$

Because S is a finite set, there is n such that $X_n = X_{n+1}$ and thus $Cl_u(T_0) = \bigcup_{k=0}^{n} X_k$ ([3]).

Definition 2. *A labelled stratified graph $\mathcal{G}$ over G is a tuple (G, L, T, u, f) where*

- *$G = (S, L_0, T_0, f_0)$ is a labelled graph*
- *$L \in Initial(L_0)$*
- *$u \in R(prod_S)$ and $T = Cl_u(T_0)$*
- *$f : (L, \{\sigma_L\}) \longrightarrow (2^{S\times S}, \{u\})$ is a morphism of partial algebras such that $f_0 \prec f$, $f(L) = T$ and if $(f(x), f(y)) \in dom(u)$ then $(x, y) \in dom(\sigma_L)$*

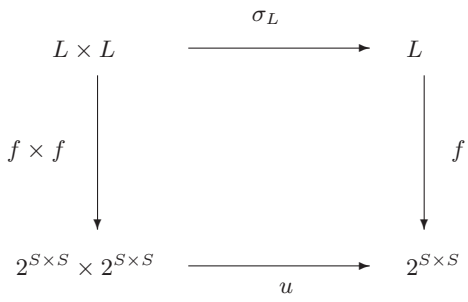

Fig. 2. The morphism condition

The last condition imposed in the definition of LSG can be interpreted as follows. The mapping f is a morphism of partial algebras, therefore in the diagram of figure 2 we have the following property: if we are able to go along the path $(L \times L, L, 2^{S\times S})$ then we are able also to go along the path $(L \times L, 2^{S\times S} \times 2^{S\times S}, 2^{S\times S})$ and we obtain the same result.

We observe now that the definition of a LSG gives the converse condition: if we can go along the path $(L \times L, 2^{S\times S} \times 2^{S\times S}, 2^{S\times S})$ then we can go along the path $(L\times L, L, 2^{S\times S})$ and we obtain the same final result. Indeed, if $(f(x), f(y)) \in dom(u)$ then $(x, y) \in dom(\sigma_L)$ therefore $\sigma_L(x, y) \in L$. Now by the morphism property we have $f(\sigma_L(x, y)) = u(f(x), f(y))$.

Based on these two remarks we observe that

$$dom(\sigma_L) = \{(x, y) \in L \times L \mid (f(x), f(y)) \in dom(u)\} \tag{3}$$

3 A general algorithm for automated drawing of a labelled graph associated to a text description

In this section we will present the manner in which a knowledge piece given in a natural language (English) can be transposed in a labelled graph. To make possible such a representation we shall suppose the knowledge piece describes several objects and some binary relations between them.

In order to extract the relations contained in the input text we have to define the syntax of the language, based on which we can also verify the correctness of the text description. We choose a mechanism based on a *Recursive Transition Network*(RTN) because the $RTNs$ are modular and very easy to maintain. They are based on *Basic Transition Networks* BTN. These networks give us a graphical representation of the behaviour of some kind of finite state automaton. So the $BTNs$ can be thought of as directed labelled graphs. The nodes of the graph are the states of the automaton. Unlike $BTNs$, the arcs of $RTNs$ can be

labelled with a command *JUMP node*. The node appearing in such a command must be a *start node* of a subnetwork. When an arc labelled with this command is traversed, a jump will be made to the subnetwork that has as initial node the node specified in the JUMP command. This network will be traversed and finally, only when it is reached its final state, the transition will continue from the point it was made the jump. This idea of breaking up a complex network into some smaller networks, one for every syntactic category (such as sentence, noun phrase, verb phrase, etc.) helps us to obtain a better structure for our linguistic knowledge. Thus, all the information about a category are contained in a corresponding subnetwork. This facilitates any syntactical modification in the language we want to define.

After we establish the desired RTN, we have to determine the binary relations that are contained in a given knowledge piece KP. When all the relations are obtained we can draw the directed labelled graph $G = (S, L_0, T_0, f_0)$. The set S of nodes is composed from the objects of the knowledge piece and the elements of L_0 are labels of binary relations. The elements of T_0 are the initial binary relations, specified in KP. The surjective mapping $f_0 : L_0 \longrightarrow T_0$ establishes the connection between labels and binary relations.

We will start the drawing of nodes with a node that has a maximum number of children and a minimum number of parents. The line of this first node will be considered the first line of the graph. The same for the column of the first node. The rest of the nodes will be drawn using the following algorithm:

1. $current\_node \leftarrow first\_node$
2. $current\_line \leftarrow first\_line$
3. $current\_column \leftarrow first\_column$
4. WHILE (we still have not drawn all the nodes)
   - 4.1 IF ($current\_node$ has child)
     - 4.1.1 we will prefer that node, named $newNode$, that accomplishes the most conditions from below:
       - 4.1.1.1 it is child of $current\_node$
       - 4.1.1.2 {parents of $newNode$ } $\cap$ { children of $current\_node$} $= \emptyset$
       - 4.1.1.3 the $current\_node$ is a child of $newNode$
     - 4.1.2 $current\_column \leftarrow current\_column + 1$
     - 4.1.3 draw $newNode$ on the cell ($current\_line$, $current\_column$)
     - 4.1.4 the $current\_node$ will be connected with the $newNode$ by a straight arc.
     - 4.1.5 delete $newNode$ from the set of children of $current\_node$
     - 4.1.6 $current\_node \leftarrow newNode$
   - 4.2 ENDIF
   - 4.3 IF ($current\_node$ does not have any child)
     - 4.3.1 $current\_line \leftarrow current\_line + 1$; $current\_column \leftarrow current\_column - 1$
     - 4.3.2 IF ($current\_node$ has a drawn parent)
       - 4.3.2.1 $current\_node \leftarrow$ the last drawn parent of $current\_node$
     - 4.3.3 ELSE

4.3.3.1 that means we have now to draw the nodes that have as children the already drawn nodes
4.3.3.2 we will choose as *current_node*, a node with most of the children already drawn
4.3.4 ENDIF
4.4 ENDIF
5. ENDWHILE

After all the nodes are drawn, we have to do the same with the arcs. We divide the arcs into two categories:

- straight arcs: these arcs are oriented from left to right and connect a node with a child found in the *IF* sentence at line 4.1 (for example, in Figure 8 the arc from $p2$ to $p4$ labelled by $B + L$)
- broken arcs: these arcs connect two nodes going around the nodes of the graph (for example, in Figure 8 the arc from *one_line* to $p4$ labelled by A)

4 From text description to image synthesis

In this section we shall present an application to reconstruct an image from a text description. We consider the following problem:

We suppose we have a remote communication line from S to R. S is the sender and R is the receiver. Let's suppose at the node S we have a chess-board of only 5×5 *squares and five pawns of the same colour. We say that a pawn* P_1 *captures the pawn* P_2 *if* P_1 *and* P_2 *are aligned on the same diagonal of the board at one square distance. We suppose that in S we see an arrangement of these pieces such that no pawn captures any other pawn. The problem is to transmit to R a text description of the scene such that R is able to reconstruct the image from S.*

In order to solve this problem we try to use in R a text analyzer, which is able to understand the text received from S and to reconstruct the image on R. This program is a *text to image converter*. We suppose the following:

- We dispose of a free-error transmission.
- The description does not specify exactly the position of a pawn, that is, the use of a sentence of the form *a pawn is on the line i and column j* is forbidden.
- Each sentence describes the position of some pawn with respect to some other pawn or the position of some empty line (column).

The description that program gets as input can be seen as a knowledge piece KP of some world of objects. Obviously, the knowledge piece will contain at least five objects, the pawns denoted by P_1, ... , P_5. Thus, the set S of objects will include the set of pawns, $S \supseteq \{P_1, ..., P_5\}$.

In order to help the program to find the positions of the pawns, the knowledge piece must contain five sentences that include binary relations between pawns. Let's denote by $Sen_1, ..., Sen_5$ five sentences of the KP, where Sen_i appears in

the input description before Sen_{i+1}, $1 \leq i \leq 4$. A good description must allow us to obtain information about the position of any pawn on the board.

Before to give a formalism of this problem we introduce the following notation: if Sen_k gives a description of the relation between P_i and P_j then we note: $Sen_k = (P_i, P_j)$. We suppose $Sen_1 = (P_{k_1}, P_{k_2})$, $Sen_2 = (P_{k_2}, P_{k_3})$, $Sen_3 = (P_{k_3}, P_{k_4})$, $Sen_4 = (P_{k_4}, P_{k_5})$ and $Sen_5 = (P_{k_5}, P_{k_1})$ where $1 \leq k_i \leq 5$ and k_i are distinct values. In this way, the sentences $Sen_1, ..., Sen_5$ offer a maximum number of information about the objects $P_1, ..., P_5$. Thus, if we transpose the relations of these five sentences in a directed labelled graph, we obtain a cycle that permits us to get new information about any two nodes using deduction process (see Figure 3).

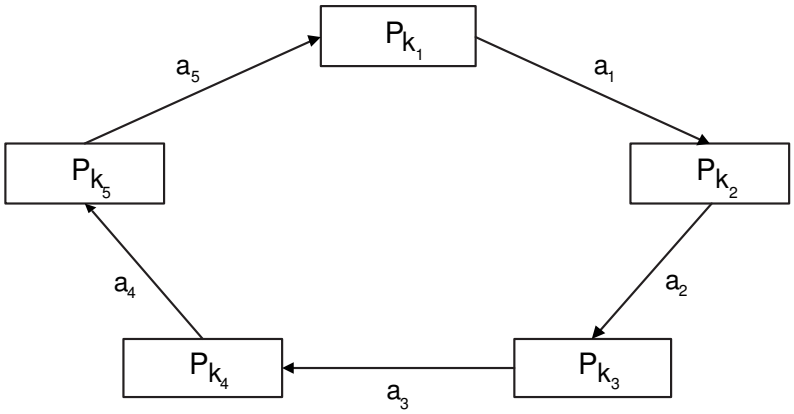

Fig. 3. A cycle of pawns

Each relation of these five sentences indicates the position of a pawn regarding some other pawn. Before enumerate the possible positions we make the following observation: a square is uniquely defined by its coordinates on the board, the pair $(line, column)$ where $1 \leq line, column \leq 5$. Thus, instead of saying that the pawn P is on the line l and column c, we will say that the square of the pawn P is (l, c).

Finally, the possible positions of a pawn are:

- **above**: a pawn P_m is above another pawn P_n if the square of P_m is (i,j) and the square of P_n is (k,j) with $k < i$.
- **below**: a pawn P_m is below another pawn P_n if the square of P_m is (i,j) and the square of P_n is (k,j) with $k > i$.
- **left side**: a pawn P_m is at left side of another pawn P_n if the square of P_m is (i,j) and the square of P_n is (i,k) with $j < k$.
- **right side**: a pawn P_m is at right side of another pawn P_n if the square of P_m is (i,j) and the square of P_n is (i,k) with $j > k$.

- **above at left/right** side: a pawn P_m is above at left (right) side of another pawn P_n if the square of P_m is (i,j) and the square of P_n is (p,k) with $i > p$ and $j < k$, respectively $k < j$.
- **below at left/right side**: a pawn P_m is below at left (right) side of another pawn P_n if the square of P_m is (i,j) and the square of P_n is (p,k) with $i < p$ and $j < k$, respectively $k < j$.

These positions are exemplified in Figure 4.

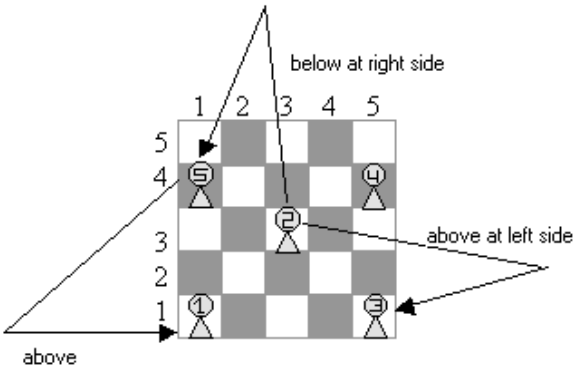

Fig. 4. Example of pawn position

We may introduce in the knowledge piece some relations that indicate how many empty lines or columns are next to some pawn. These relations are not compulsory. They make the description of the positions of the pawns more accurate. If the knowledge piece contains such relations then the set S of the objects will include a reference to such line(s) or column(s).

The description can be made in a natural language. To define the syntax of the language we have constructed the following RTN. As can be seen in Figure 5, the RTN is composed by eight subnetworks: the subnetwork S defines the syntax of a sentence in our language, the Np_sg and Np_pl are subnetworks for noun phrase, Vp_sg and Vp_pl are subnetworks for verb phrase, $Place$ realizes a connection between some subnetworks, Ap is subnetwork for adverb phrase and the $Side$ subnetwork defines the structure of the side entities.

In what follows we will give an example of an arrangement and a possible knowledge piece that describe it:

The pawn P_3 is at right side of P_1. P_1 is below P_2 at left side. P_2 is below P_4 at left side. P_4 is at right side of P_5. P_5 is above P_3 at left side. An empty line is below P_2. An empty line is above P_4.

The labelled graph illustrating this piece of knowledge is $G = (S, L_0, T_0, f_0)$, where:

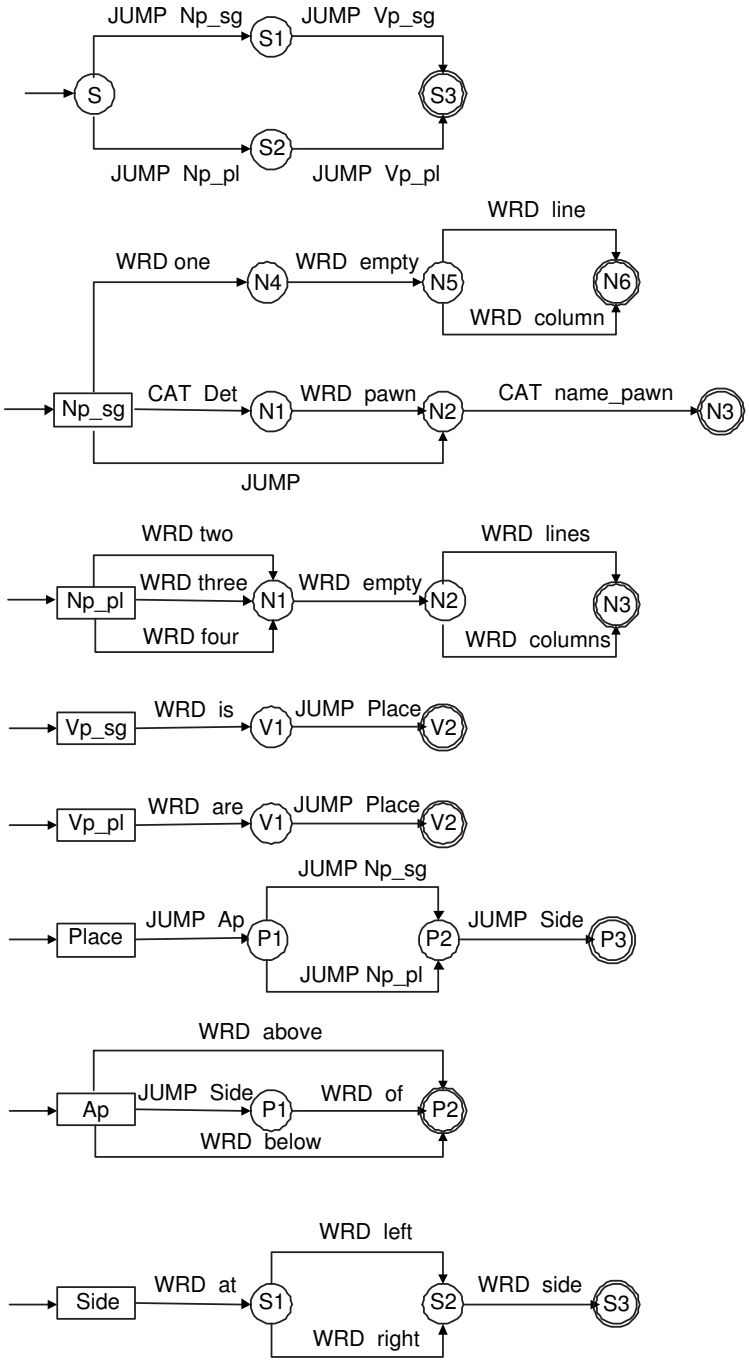

Fig. 5. The syntax defined by a RTN

- the set of objects is $S = \{P_1, P_2, P_3, P_4, P_5, one\_line\}$
- the set of the binary relations is $T_0 = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\}$ and:
  $\rho_1 = \{(P_3, P_1), (P_4, P_5)\}, \rho_2 = \{(P_1, P_2), (P_2, P_4)\}, \rho_3 = \{(P_5, P_3)\}$,
  $\rho_4 = \{(one\_line, P_2)\}, \rho_5 = \{(one\_line, P_4)\}$
- the set of labels $L_0 = \{R, B, A, B + L, A + L\}$, where:
  - the symbol R describes the semantics *is at right side of*
  - the symbol B is for the meaning *is below*
  - the symbol A is for the meaning *is above*
  - the symbol $B + L$ is associated to *is below at left side*
  - the symbol $A + L$ is associated to *is above at left side*
- the function $f_0 : L_0 \longrightarrow T_0$ is a bijective mapping because for each label e of L_0 there is an unique binary relation $r \in T_0$ such that $f_0(e) = r$:

$$f_0(R) = \rho_1, f_0(B) = \rho_4, f_0(A) = \rho_5, f_0(B + L) = \rho_2, f_0(A + L) = \rho_3$$

The labelled graph obtained is represented in Figure 6.

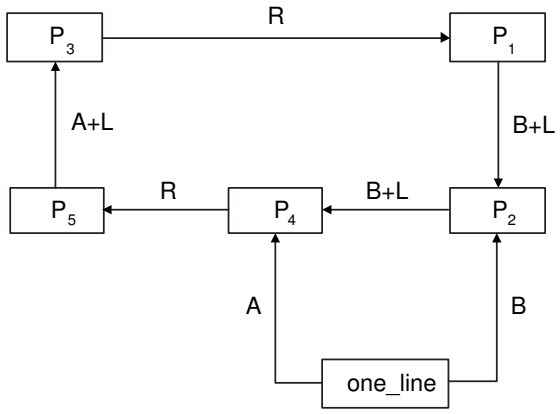

Fig. 6. Labelled graph of KP

In order to obtain a labelled stratified graph ([3]) the following computations are realized:

- $T_0 = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5\}$, where
  $\rho_1 = \{(P_3, P_1), (P_4, P_5)\}, \rho_2 = \{(P_1, P_2), (P_2, P_4)\}, \rho_3 = \{(P_5, P_3)\}$,
  $\rho_4 = \{(one\_line, P_2)\}, \rho_5 = \{(one\_line, P_4)\}$
- $T_1 = T_0 \cup \{\rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}, \rho_{11}\}$, where
  $\rho_6 = u(\rho_2, \rho_2) = \{(P_1, P_4)\}$, $\rho_7 = u(\rho_2, \rho_1) = \{(P_2, P_5)\}$,
  $\rho_8 = u(\rho_1, \rho_3) = \{(P_4, P_3)\}$, $\rho_9 = u(\rho_3, \rho_1) = \{(P_5, P_1)\}$,
  $\rho_{10} = u(\rho_1, \rho_2) = \{(P_3, P_2)\}$, $\rho_{11} = u(\rho_5, \rho_1) = \{(one\_line, P_5)\}$
  $\rho_5 = u(\rho_4, \rho_2)$

- $T_2 = T_1 \cup \{\rho_{12}, \rho_{13}, \rho_{14}, \rho_{15}, \rho_{16}, \rho_{17}\}$, where
  $\rho_{11} = u(\rho_4, \rho_7)\}$, $\rho_{12} = u(\rho_5, \rho_8) = u(\rho_{11}, \rho_3) = \{(one\_line, P_3)\}$,
  $\rho_{13} = u(\rho_{11}, \rho_9) = \{(one\_line, P_1)\}$, $\rho_{14} = u(\rho_6, \rho_1) = \{(P_1, P_5)\}$,
  $\rho_{15} = u(\rho_8, \rho_1) = \{(P_4, P_1)\}$, $\rho_{16} = u(\rho_{10}, \rho_2) = \{(P_3, P_4)\}$,
  $\rho_{17} = u(\rho_{10}, \rho_7) = \{(P_3, P_5)\}$
- We append now
  $\rho_{13} = u(\rho_{12}, \rho_1) = u(\rho_5, \rho_{14})$

Now we can compute the set L and the values of the mapping $f : L \longrightarrow T$, where $T = T_2$:

- $f(R) = f_0(R) = \rho_1$; $f(B) = f_0(B) = \rho_4$; $f(A) = f_0(A) = \rho_5$;
  $f(B + L) = f_0(B + L) = \rho_2$; $f(A + L) = f_0(A + L) = \rho_3$;
  $L_1 = L_0 \cup \{B + L, A + L\}$
- $f(\sigma(B + L, B + L)) = \rho_6$; $f(\sigma(B + L, R)) = \rho_7$;
  $f(\sigma(R, A + L)) = \rho_8$; $f(\sigma(A + L, R)) = \rho_9$;
  $f(\sigma(R, B + L)) = \rho_{10}$; $f(\sigma(A, R)) = \rho_{11}$;
  $f(\sigma(B, B + L)) = \rho_5$ $L_2 = L_1 \cup \{\sigma(B + L, B + L), \sigma(B + L, R)$,
  $\sigma(R, A + L), \sigma(A + L, R), \sigma(R, B + L), \sigma(A, R), \sigma(B, B + L)\}$;
- $f(\sigma(B, \sigma(B + L, R))) = \rho_{11}$;
  $f(\sigma(A, \sigma(R, A + L))) = f(\sigma(\sigma(A, R), A + L)) = \rho_{12}$;
  $f(\sigma(\sigma(A, R), \sigma(A + L, R))) = \rho_{13}$; $f(\sigma(\sigma(B + L, B + L), R)) = \rho_{14}$;
  $f(\sigma(\sigma(R, A + L), R)) = \rho_{15}$; $f(\sigma(\sigma(R, B + L), B + L)) = \rho_{16}$;
  $f(\sigma(\sigma(R, B + L), \sigma(B + L, R))) = \rho_{17}$;
  $L_3 = L_2 \cup \{\sigma(B, \sigma(B + L, R)), \sigma(A, \sigma(R, A + L)), \sigma(\sigma(A, R), A + L)$,
  $\sigma(\sigma(A, R), \sigma(A + L, R)), \sigma(\sigma(B + L, B + L), R), \sigma(\sigma(R, A + L), R)$,
  $\sigma(\sigma(R, B + L), B + L), \sigma(\sigma(R, B + L), \sigma(B + L, R))\}$
- $f(\sigma(\sigma(\sigma(A, \sigma(R, A + L))), R)) = f(\sigma(\sigma(\sigma(\sigma(A, R), A + L)), R)) =$
  $f(\sigma(A, \sigma(\sigma(B + L, B + L), R))) = \rho_{13}$;
  $L = L_3 \cup \{\sigma(\sigma(\sigma(A, \sigma(R, A + L))), R)), \sigma(\sigma(\sigma(\sigma(A, R), A + L)), R))$,
  $\sigma(A, \sigma(\sigma(B + L, B + L), R))\}$

Now we obtained the labelled stratified graph $K = (G, L, T, u, f)$ over $G = (S, L_0, T_0, f_0)$.

In what follows we will give a short presentation of the algorithm we used in order to obtain the right positions of the pawns. It is based on the binary relations of the set T (the set T is also named *the environment for reasoning*). Before to present the algorithm we have to specify the notations we used in:

- for every pawn P_i with $1 \leq i \leq 5$ we note by l_i and c_i the line, respectively the column, of the pawn P_i.
- ρ_p is a *below relation* if there exists $l \in L$ such that $\rho_p = f(l)$ and l is a *below label*. We define the below labels as follows:
  - $B, B + L, B + R$ are below labels
  - $\sigma(X, Y)$ is a below label for every below labels X and Y
  - $\sigma(X, Y)$ is a below label for every below label X and $Y \in \{L, R\}$

- ρ_p is a *left relation* if there exists $l \in L$ such that $\rho_p = f(l)$ and l is a *left label*. We define the left labels as follows:
  - $L, A + L, B + L$ are left labels
  - $\sigma(X, Y)$ is a left label for every left labels X and Y
  - $\sigma(X, Y)$ is a left label for left label X and $Y \in \{A, B\}$
- we have $l_i < l_j$ if there exists a below relation $\rho_p \in T$ such that $(P_i, P_j) \in \rho_p$.
- we have $c_i < c_j$ if there exists a left relation $\rho_p \in T$ such that $(P_i, P_j) \in \rho_p$.

First of all we determine the lines of the pawns. Then we determine the columns of these pieces taking care that for every pawn P_i there must not exist a pawn P_j such that $|l_i - l_j| = 1$ and $|c_i - c_j| = 1$. Otherwise the pawns could capture between them.

The algorithm to obtain the lines of the pawns is the following:

1. $current\_line \leftarrow 1$ and $Det\_pawns \leftarrow \emptyset$
2. WHILE $card(Det\_pawns) < 5$

   2.1 for all pawns P_i, $l_i \leftarrow current\_line$ if for every $P_j \in Det\_pawns$ we have $l_j < l_i$

   2.2 $Det\_pawns \leftarrow Det\_pawns \cup \{P_i\}$ and

   $$current\_line \leftarrow current\_line + 1$$

3. ENDWHILE
4. If the KP contains empty line(s)-pawn relations, some lines obtained above will be shifted up, if is necessary, in order to accomplish these relations.

where $card(X)$ denotes the cardinal number of the set X.

The algorithm to obtain the columns of the pawns is:

1. $current\_col \leftarrow 1$ and $Det\_pawns \leftarrow \emptyset$
2. WHILE $card(Det\_pawns) < 5$

   2.1. for all pawns P_i, $c_i \leftarrow current\_col$ if for every $P_j \in Det\_pawns$ we have $c_j < c_i$.

   Let P the set of all these pawns P_i. If for some pawn $P_i \in P$ there is a pawn $P_j \in Det\_pawns$ such that $|l_j - l_i| = 1$ and $|c_j - c_i| = 1$ then we delete P_i from P.

   2.2. $Det\_pawns \leftarrow Det\_pawns \cup P$

   2.3. $current\_col \leftarrow current\_col + 1$.

3. ENDWHILE
4. If the KP contains empty column(s)-pawn relations, some columns obtained above will be shifted to the right, if is necessary, in order to accomplish these relations.

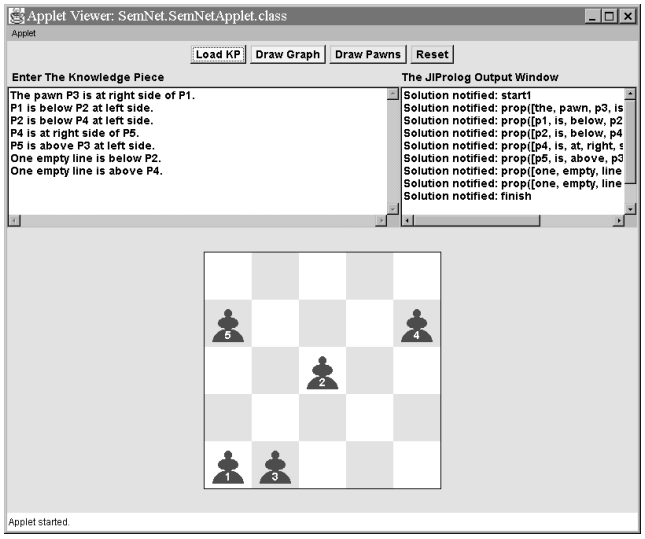

Fig. 7. The first window of the applet

5 Implementation

We have implemented the presented theory in an application that consists of a Java applet and a Prolog file. In this application we used the JIProlog (JavaInternetProlog) product written by Ugo Chirico ([5]) in order to establish a connection between the Java applet and the Prolog file.

The *Prolog file* uses the RTN presented above in Figure 5 in order to divide each input sentence in the main syntactic categories. Thus, it is easy to obtain the objects and the binary relation between them. These information are the output of the Prolog file for every input sentence that respects the syntax defined in the implemented RTN.

The *Java applet* consists of two windows:

- the first window appears when the applet is loaded. Using the controls of this window the user can:
  - enter the Knowledge Piece
  - "load" the Knowledge Piece introduced by pressing the *Load KP* button. As a result, using the JIProlog connection, the Java applet will send all the sentences of the KP to the Prolog file in order to obtain back the relations included.
  - see the positions of the pawns determined by the application according with the KP introduced and loaded successfully.
    In the figure 7 the solution of the program for the KP presented in the section 4 can be seen. This solution approach very much to the initial image illustrated in Figure 4 with one difference: the column of

the pawn P_3 is not the right one. This can be explained by the fact that the *reasoning environment* T does not give very much information about the column of this pawn.

- the second window appears when the *DrawGraph* button of the first window is pressed. This window contains a group of *Color buttons*, a canvas in center and another group of buttons: *Movement buttons*.
  The canvas is used for drawing the labelled graph of the relations obtained by the Prolog file. The *Color buttons* were introduced in the application in order to point out one node and together with it all the arcs that have it as start node. This is done by drawing the chosen node an the arcs with a different color. The other group of buttons can be used for moving the graph inside the canvas.
  Because the solution we have implemented for drawing the graph does not guarantee that there is not any crossing arcs, we also gave the possibility of rearranging the graph by dragging the nodes (and together with them all the arcs that start from them are redrawn) and the corners of the broken arcs in the desired position.

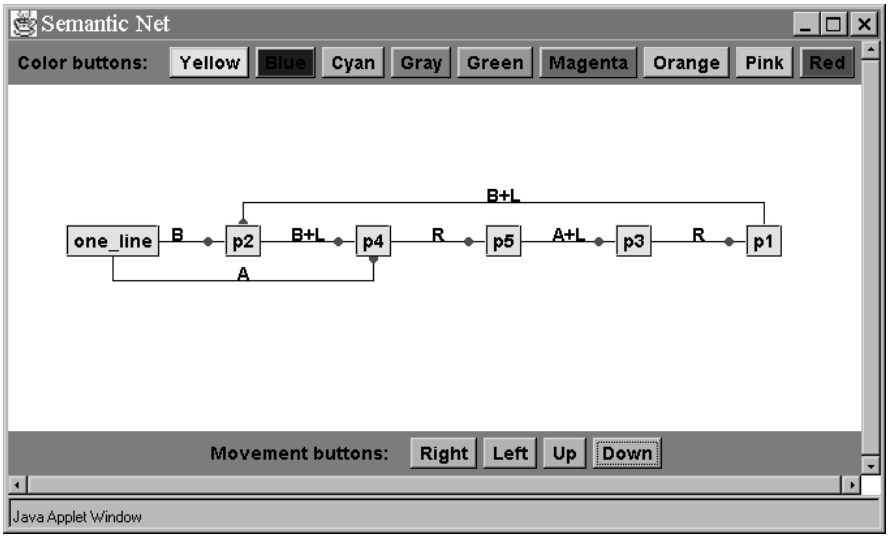

Fig. 8. The second window of the applet

6 Conclusions and future work

An application of the labelled stratified graphs in image synthesis is presented. Based on a recursive transition network, a given text is analyzed and its

information is extracted. Some labelled stratified graph is constructed and this structure permits us to obtain the image described in the text. If the text does not contain a complete description then a near exact image is obtained. This is illustrated in the example presented in the previous sections. In a future work we shall study the ambiguity of the text and its implications in image synthesis.

References

[1] Leonard Bolc (Ed.), The Design of Interpreters, Compilers and Editors for Augmented Transition Network, Springer-Verlag, 1983
[2] Ţăndăreanu N. (2000), Knowledge Bases with Output, Knowledge and Information Systems, 2, 438-460.
[3] Ţăndăreanu N. (2000), Proving the existence of labelled stratified graphs, Annals of the University of Craiova, Mathematics and Computer Science Series, XXVII, 81-92; on line version at http://inf.ucv.ro/~ntand/en/publications.html
[4] Ţăndăreanu N. (2001), Expert Systems, Knowledge representation and inference, Universitaria Publishing House
[5] Ugo Chirico: http://www.ugosweb.com