## THE SYNTHESIS OF A COMPUTER KNOWLEDGE BASE

## SYNTEZA BAZY WIEDZY KOMPUTERA


#### Abstract

Users' program contain the enormous output, the heritage of human knowledge. This knowledge appears in a structure of a program but also in the sequence of operation written in this structure. In this work it is shown such of organizing a base of knowledge, which lets use evolution software to do a synthesis of the user's program. The representation of knowledge is presented in the form of graphs, witch explicitly build up a logical structure of the hardware.

Keyword: knowledge, structure of a program, symbolic expressions, evolvable hardware

\section*{Streszczenie}

Komputery rozwiązują różne problemy, w których mieści się olbrzymi dorobek wiedzy ludzkiej. Wiedza ta zawarta jest w strukturze programu oraz operacjach arytmetycznych i logicznych wpisanych $w$ tę strukture. Baza wiedzy takiego komputera powinna być wyrażona w odpowiedniej postaci symbolicznej, zapewniającej maksymalną integrację wiedzy w jego pamięci. Celem artykułu jest określenie sposobu transformacji programów użytkownika na postać symboliczną. Sformalizowano w nim koncepcję nabywania wiedzy zawartej w komentarzach, zdaniach deklarujących, strukturze algorytmu i sekwencjach kodów operacji. Słowa kluczowe: baza wiedzy, struktura programu, wyrażenie symboliczne, ewolucyjny


 hardware[^0]
## 1. Introduction

A computer can solve different problems. The way they are solved in proves a vast human knowledge. If the computer does not acquire this knowledge, it does not draw any conclusions from what it has done, then it does not learn - thus the question arises whether it is possible to build a computer which would be capable of learning, or in other words of acquiring knowledge from the user's programmes [1]. Such an ability is characteristic of the computers with adequate organization of the knowledge basis. The knowledge basis of such a computer should be expressed in an adequate symbolic form [1].

In this paper we develop that approach in domain of automatic synthesis of user program from small pieces of knowledge stored in the knowledge base and acquired from another different programs.

The above mentioned approach to automatic programming is such that on the basis of user's requirement expressed in natural language the computer performs the synthesis of a program to solve the user's task mentioned in the requirement. In order to accomplish that approach, the computer possesses suitable knowledge in its knowledge base.

These programs are purposely entered into the computer by the knowledge engineer to construct the initial knowledge base. During performance of the automatic programming system the initial knowledge is extended by computer in result of consultation with user. During acquisition of knowledge, the entered program is split by the computer into four parts: sequence of comments, algorithm structure, sequence of operation codes and declaration. Each such a part is split by the computer into some components. These components are treated as small computer into symbolic forms which forms which are incorporated into the knowledge base.

Each knowledge element is represented only once in the computer memory, no matter how many programmes this element will occur in. Such an attitude ensures the integration of knowledge. For its realisation it was assumed that the knowledge basis is represented in the computer by tables any symbolic expressions. In the Paper the rules according to which knowledge is acquired from the user's programme is presented. Next, there is shown the way of expanding of the symbolic expression representing the knowledge basis by adding the knowledge gained from the user's programmes and making conclusions on the basis of this knowledge. Such an attitude demands determining all the connections of the elements in the knowledge basis. It is a complex problem and demands deep analysis.

The leading idea of the resented approach to automatic programming arises from two facts. The first fact tells us that in each users program written in a programming language there is included human knowledge about a problem and method of solving it. We well know that the present computer does not remember the knowledge included in user program. When executing a given user program is completed by the computer, the program is erased from computer memory. Hence, after erasing the user program, the computer does not know what problem was solved and does not remember the human knowledge that was concealed in the executed program. According to the above remarks, the computer should acquire the knowledge from user programs to own knowledge base. The second fact tells us that user programs written in programming language consist of finite number of components such as structural components of algorithm, groups of structural components, structural components of declaration, kinds of operation codes and so on. In different programs the same components can appear. The above mentioned components can be

BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ
considered as small pieces of knowledge which, after transforming into symbolic form, should be stored in the knowledge bases. From those pieces of the knowledge a user program con be constructed [7].

## 2. General concept of user's programme decomposition into elements

Each algorithm consists of two base parts: the description of objects on which it operates, and the description of activities performed on these objects. The objects are the data which appear in arithmetical or logical expressions. Specified activities described by means of an instruction of a chosen language of programming are performed on these objects. A programme of higher abstract level contains:

- Comments - comments are written in the natural language. They contain information about problems that can be solved by a computer;
- Declaring part - consists of a set of definitions and declarations of various types. Each of the mentioned sets can be omitted or can appear many times and in any order;
- Structural part - contains programming language instructions connected with each other into a logical whole. Such instructions performed in a specified order make a given problem possible to solve. Figure 1 shows the user's programme decomposition into elements;
- Operational part - arithmetical and logical operations are inscribed into the algorithm structure.
The knowledge included in the user's programme appears in its structure and in a sequence of operations written into this structure. Declaring sentences are associated with the operational part of the algorithm, they allow correct interpretation of all changeable occurring there. A computer should acquire this knowledge and integrate it into its knowledge basis. Therefore, to enable the computer to collect and use this knowledge it is essential to have it included in the knowledge basis. This knowledge should be presented in a form that would make it possible to be reconstructed. Having the above in mind, this knowledge is expressed in conditional rules of the IF THEN type, which are then transformed into symbolic expressions. Symbolic expressions, in turn, should have such a form that would make them possible to be transformed in a computer. All the operations in the knowledge basis are performed on symbolic expressions [4].

To realize the process of acquiring knowledge from the user's programme, the computer should possess an adequate software. It is assumed that the knowledge contained in the first user's programme is the initial knowledge basis. The initial knowledge basis is enlarged by the knowledge acquired from other user's programmes. The acquired knowledge overleaps with the knowledge already included in the knowledge basis. It is done by comparing symbolic expressions of the knowledge acquired with those which represent the hitherto existing knowledge basis. If the compared part of the symbolic expression does not exist in the knowledge basis, then the expression representing the knowledge basis will be enlarged by this part. Otherwise the knowledge basis remains unchanged. The acquired knowledge is integrated, that is the same knowledge elements are all remembered in the same place of the computer memory. To determine the knowledge basis, as well as adequate programme solutions which would ensure the user's programmes synthesis are the most crucial problems in automatic programming.


Fig. 1. Illustration of the process of acquiring knowledge from the user's programme
Rys. 1. Ilustracja procesu nabywania wiedzy z programów użytkownika

## 3. Decomposition user's programme

### 3.1. The first user's commentaries decomposition

We assume that the base of knowledge is empty. The first program introduced to the computer has such shape

## Multiply matrix A by matrix B

The natural language words in this commentary are substituted by symbolic names $b j$ $(j=1,2,3, \ldots)$ which are remembered in table 1 [3].

Table 1
Vocabulary B

| Symbolic names | Adress in ${K_{\text {ext }}{ }^{++}}^{\text {Words }}$ |  |
| :---: | :---: | :---: |
| $b_{1}$ | $W_{1}$ | Multiply |
| $b_{2}$ | $W_{4}$ | matrix |
| $b_{3}$ | $W_{7}$ | $\boldsymbol{A}$ |
| $b_{4}$ | $W_{10}$ | $\boldsymbol{b y}$ |
| $b_{5}$ | $W_{12}$ | $\boldsymbol{B}$ |

In the next stage of the analysis the task is divided into groups of words. The first group refers to the predicate, the second to the subject and the third one to the object. The words appearing in sentences belong to different lexical categories. Each word in the vocabulary has a given lexical category. In the analyzed sentence it will be a set $b_{1} b_{2} b_{3} b_{4} b_{5}$. The computer divides this set into word groups, making use of lexical symbols set, according to the rules given in [5]. Each group of words has a different code symbol $g_{i}(i=1,2,3, \ldots)$

$$
\begin{array}{lllll}
b_{1} & \rightarrow & g_{1} & \rightarrow & \text { Multiply } \\
b_{2} b_{3} & \rightarrow & g_{2} & \rightarrow & \text { matrix } \boldsymbol{A} \\
b_{4} b_{2} b_{5} & \rightarrow & g_{3} & \rightarrow & \text { by matrix } \boldsymbol{B}
\end{array}
$$

A symbolic expression is a form of knowledge integration. Each word group is analysed separately. For each group independent symbolic expressions are formed

$$
\begin{gather*}
B^{+}\left(g_{1}\right)={ }^{0}\left(b_{1}^{1}\left(z_{4} g_{1} b_{1}\right)^{1}\right)^{0}  \tag{1}\\
B^{+}\left(g_{2}\right)={ }^{0}\left(b_{2}^{1}\left(z_{1} g_{2} b_{3}^{2}\left(z_{4} g_{2} b_{2}\right)^{2}\right)^{1}\right)^{0}  \tag{2}\\
B^{+}\left(g_{3}\right)={ }^{0}\left(b_{4}{ }^{1}\left(z_{1} g_{3} b_{2}^{2}\left(z_{1} g_{3} b_{5}^{3}\left(z_{4} g_{3} b_{4}\right)^{3}\right)^{2}\right)^{1}\right)^{0} \tag{3}
\end{gather*}
$$

Symbol $z_{1}$ stands for the space between symbols $b_{j}$ and symbol $z_{4}$ means a connection of the final symbol to the initial one. All the symbolic expressions are united, thus forming a comprehensive expression has the form

$$
\begin{gather*}
B_{1}^{++}=B^{+}\left(g_{1}\right) \oplus B^{+}\left(g_{2}\right) \oplus B^{+}\left(g_{3}\right)  \tag{4}\\
B_{1}^{++}={ }^{0}\left(b_{1}^{1}\left(z_{4} g_{1} b_{1}\right)^{1}, b_{2}^{1}\left(z_{1} g_{3} b_{5}, z_{1} g_{2} b_{3}{ }^{2}\left(z_{4} g_{2} b_{2}\right)^{2}, b_{4}{ }^{1}\left(z_{1} g_{3} b_{2}\right)^{1}, b_{5}{ }^{1}\left(z_{4} g_{3} b_{4}\right)^{1}\right)^{0}\right. \tag{5}
\end{gather*}
$$

$\oplus$ is operation of resting knowledge.
The structure of the commentary is a linear sequence of the words groups which can be written as a graph illustrated by Fig. 2.


Fig. 2. Graphic interpretation of commentary structure Rys. 2. Graficzna ilustracja struktury komentarza

The analytical form is presented in the following symbolic expression

$$
\begin{gather*}
C_{1}^{+}={ }^{0}\left(g_{1}^{1}\left(z_{1} c_{1} g_{2}^{2}\left(z_{1} c_{1} g_{3}^{3}\left(z_{4} c_{1} g_{4}\right)^{3}\right)^{2}\right)^{1}\right)^{0}  \tag{6}\\
A_{1}^{+}={ }^{0}\left(c_{1}^{1}\left(z_{4} p_{1} c_{1}\right)^{1}\right)^{0} \tag{7}
\end{gather*}
$$

On the basis of the commentary in the user's programme there is formed a word basis of knowledge. If the user instructs the computer to solve a certain problem, the user's instruction can be compared with the commentary in the knowledge basis. The result of comparison is a decision, elaborated by the computer, about the possibility of solving the given task.

### 3.2. The first user's in declaration part

Each procedure written in the language of higher level has to have univocally determined all variables that appear in it. These are the definitions of constants, definitions of the type of variables and declarations of variables. They may be neglected or appear repeatedly and in any order. The declaration part from the first user's programme, which allows has the following form [3-5]
(The first form of declaration)

```
const \(\boldsymbol{n}=50\);
type = array [l..n, 1..n] of real;
var \(i, j, k: 1 . . n ;\)
A, B, C: M;
\(r:\) real;
```

In order to standardize changeable names, virtual symbols have been introduced instead of real changeable names.
(The second form of declaration)

```
const \(v=\chi ;\)
type \(w=\) array [ \(1 . . v, 1 . . v]\) of real;
var s, s, s:w;
\(s, s, s: w ;\)
s: real;
```

In agreement with the declaration part transforming modulus, this part is then divided into a sequence of terms - called declaration components. Giving to the declaration components their symbolic names $k_{i}$ takes place on the bases of decisive rules. In the case analyzed these are the rules

```
IF const \(v=x\) THEN \(k_{1}\)
IF type \(w=\) array [1.. v, 1.. v] of real THEN \(k_{2}\)
IF var \(s, s, s: w\) THEN \(k_{3}\)
IF \(s, s, s: w\) THEN \(k_{4}\)
IF \(s:\) real THEN \(k_{5}\)
```

Having gained symbols $k_{i}$, the declaration components are then subjected to transformation into symbolic expression. Ascribing symbols $l_{i}$ to the words of statement components remembered in vocabulary $L$.

Table 2
Vocabulary $L$

| Symbolic names | Adress in $K_{\text {ext }}{ }^{++}$ | Words |
| :---: | :---: | :---: |
| $l_{1}$ | $V_{1}$ | const |
| $l_{2}$ | $V_{6}$ | type |
| $l_{3}$ | $V_{20}$ | $\boldsymbol{v a r}$ |
| $l_{4}$ | $V_{3}$ | $\boldsymbol{v}=\boldsymbol{x}$ |
| $l_{5}$ | $V_{8}$ | $\boldsymbol{w}=$ |
| $\ldots$ | $\ldots$ | $\ldots$ |
| $l_{13}$ | $V_{31}$ | $\boldsymbol{s}:$ |

For $k_{3}$ declaration component the expression has the form

$$
\begin{gather*}
l_{3} l_{10} l_{11} \rightarrow k_{3} \\
K_{3}^{+}={ }^{0}\left(l_{3}{ }^{1}\left(z_{1} k_{3} l_{10}{ }^{2}\left(z_{1} k_{3} l_{11}{ }^{3}\left(z_{4} k_{3} l_{3}\right)^{3}\right)^{2}\right)^{1}\right)^{0} \tag{8}
\end{gather*}
$$

All the symbolic expression are united, thus forming a comprehensive expression representing a set of declaration components

$$
\begin{equation*}
K_{1}^{++}=K_{1}^{+} \oplus K_{2}^{+} \oplus K_{3}^{+} \oplus K_{4}^{+} \oplus K_{5}^{+} \tag{9}
\end{equation*}
$$

$K_{1}^{++}={ }^{0}\left(l_{1}{ }^{1}\left(z_{1} k_{1} l_{4}{ }^{2}\left(z_{4} k_{1} l_{1}\right)^{2}\right)^{1}, l_{2}^{1}\left(z_{1} k_{2} l_{5}{ }^{2}\left(z_{1} k_{2} l_{6}{ }^{3}\left(z_{1} k_{2} l_{7}^{4}\left(z_{1} k_{2} l_{8}^{5}\left(z_{1} k_{2} l_{9}{ }^{6}\left(z_{4} k_{5} l_{13}, z_{4} k_{2} l_{2}\right)^{6} \ldots\right)^{1}\right.\right.\right.\right.\right.$, $\left.l_{3}{ }^{1}\left(z_{1} k_{3} l_{10}{ }^{2}\left(z_{1} k_{3} l_{11}, z_{1} k_{4} l_{12}{ }^{3}\left(z_{4} k_{4} l_{10}\right)^{3}\right)^{2}\right)^{1}, l_{11}{ }^{1}\left(z_{4} k_{3} l_{3}\right)^{1}, l_{13}{ }^{1}\left(z_{1} k_{5} l_{9}\right)^{1}\right)^{0}$
Another very essential set is the set of declaration structures. The structure of the programme is expressed by a sequence of $V_{i}$ declaration components

$$
V_{1}=k_{1}, k_{2}, k_{3}, k_{4}, k_{5}, k_{0}
$$

$k_{0}$ is an empty declaration component. Declaration components belonging to the first user's programme belong to one particle of $t$ parameter. The symbolic expression describing the structure of the declaration part has the form

$$
V_{1}^{+}={ }^{0}\left(k _ { 1 } ^ { 1 } \left(z _ { 1 } t _ { 1 } k _ { 2 } ^ { 2 } \left(z _ { 1 } t _ { 1 } k _ { 3 } ^ { 3 } \left(z_{1} t_{1} k_{4}^{4}\left(z_{1} t_{1} k_{5}^{5}\left(z_{4} t_{1} k_{0}\right)^{5} \ldots\right)^{0}\right.\right.\right.\right.
$$

In the declaration structures there occurs only one $t$ parameter group written as

$$
T_{1}^{+}={ }^{0}\left(t_{1}^{1}\left(z_{4} p_{1} t_{1}\right)^{1}\right)^{0}
$$

### 3.3. The first user's in structural part

In the algorithm structure certain Structural components can be distinguished. When analyzing the Structural part, the subsequent instructions of the algorithm have to be considered. For the analyzed programme they have the following form:

```
begin
fori:= 1 to n do
begin
forj:= 1 to n do
begin
r:=0;
for k:=1 to n do
r:=r +A[i,k]* B{k,j]
C[i,j]:= r;
end;
end;
end;
```

All the arithmetical and logical operations are replaced by symbols op and opp. The word next informs that between the words begin and end there occur several instructions [3, 4, 6].

Rules IF - THEN which allow giving the Structural components their symbolic names as well as coefficients $U_{j}$ characterising their properties. For the analysed algorithm five
rules are chosen because five different Structural components have occurred. These rules have the following form


Having gained symbols $f_{i}$, the structural components are the subjected to transformation into symbolic expression which for the first structural component takes the following form

> IF begin next end THEN $f_{1} U_{1} U_{2}$
> IF for op do next THEN $f_{2} U_{1}$
> IF begin op next opp end $T H E N f_{3} U_{1} U_{2} U_{3}$
> IF begin op do opp THEN $f_{4} U_{1}$

Ascribing symbols $d_{i}$ to the words of statement components remembered in vocabulary $D$.
Table 3
Vocabulary D

| Symbolic names | Adress in Gext $^{\text {ex }}$ | Words |
| :---: | :---: | :---: |
| $d_{1}$ | $X_{1}$ | begin |
| $d_{2}$ | $X_{19}$ | for |
| $d_{3}$ | $X_{4}$ | $\boldsymbol{o p}$ |
| $d_{4}$ | $X_{8}$ | do |
| $d_{5}$ | $X_{11}$ | next |
| $d_{6}$ | $X_{15}$ | end |
| $d_{7}$ | $X_{23}$ | $\boldsymbol{o p p}$ |

$$
\begin{gather*}
d_{1} d_{5} d_{6} \rightarrow f_{1} U_{1} U_{2} \\
G_{1}^{+}={ }^{0}\left(d _ { 1 } ^ { 1 } \left(z_{1} f_{1} d_{5}^{2}\left(z_{1} f_{1} d_{6}^{3}\left(z_{4} f_{1} U_{1} U_{2}\right)^{3} \ldots\right)^{0}\right.\right. \tag{11}
\end{gather*}
$$

Other structural components are written in an analogous way. A comprehensive symbolic expression which represents the basis of the computer knowledge in the field of structural components is obtained after having completed the covering operation

$$
\begin{equation*}
G_{1}^{++}=G_{1}^{+} \oplus G_{2}^{+} \oplus G_{3} \oplus G_{4} \tag{12}
\end{equation*}
$$

$G_{1}^{++}={ }^{0}\left(d_{1}^{1}\left(z_{1} f_{1} d_{5}, z_{1} f_{3} d_{3}{ }^{2}\left(z_{1} f_{3} d_{5}, z_{1} f_{4} d_{4}, z_{1} f_{2} d_{4}^{3}\left(z_{1} f_{4} d_{7}, z_{1} f_{2} d_{5}^{4}\left(z_{4} f_{2} U_{1}, z_{1} f_{3} d_{7}, z_{1} f_{1} d_{6}\right.\right.\right.\right.\right.$,
$\left.\left.{ }^{5}\left(z_{4} f_{1} U_{1} U_{2}, z_{4} f_{3} U_{1} U_{2} U_{3}\right)^{5} \ldots\right)^{1}, d_{2}{ }^{1}\left(z_{1} f_{4} d_{3}, z_{1} f_{2} d_{3}\right)^{1}, d_{7}{ }^{1}\left(z_{4} f_{4} U_{1}, z_{1} f_{3} d_{6}\right)^{1}\right)^{0}$
The structural of the algorithm is presented by the symbolic expression below

$$
\begin{gather*}
F_{1}^{+}=f_{1} U_{1} U_{2}, f_{2} U_{1}, f_{1} U_{1} U_{2}, f_{2} U_{1}, f_{3} U_{1} U_{2} U_{3}, f_{4} U_{1}  \tag{14}\\
e_{1}\left(f_{1}, f_{2}\right) \quad e_{2}\left(f_{1}, f_{2}, f_{3}, f_{4}\right) \\
F_{1}^{+}=e_{1} f_{1} U_{1} U_{2}, e_{1} f_{2} U_{1}, e_{2} f_{1} U_{1} U_{2}, e_{2} f_{2} U_{1}, e_{2} f_{3} U_{1} U_{2} U_{3}, e_{2} f_{4} U_{1}  \tag{15}\\
F_{1}^{+}\left(e_{1}\right)={ }^{0}\left(f _ { 1 } ^ { 1 } \left(U _ { 1 } e _ { 1 } f _ { 2 } ^ { 2 } \left(U_{1} e_{1} f_{*}^{3}\left(U_{*} e_{1} f_{1}^{4}\left(U_{2} e_{1} f_{0}\right)^{3} \ldots\right)^{0}\right.\right.\right.  \tag{16}\\
F_{1}^{+}\left(e_{2}\right)={ }^{0}\left(U_{2} e_{2} f_{0}, U_{1} e_{2} f_{2}^{2}\left(U_{1} e_{2} f_{3}^{3}\left(U_{3} e_{2} f_{3}, U_{2} e_{2} f_{1}, U_{1} e_{2} f_{4}^{4}\left(U_{1} e_{2} f_{3}\right)^{4} \ldots\right)^{0}\right.\right.  \tag{17}\\
F_{1}^{++}=F_{1}^{+}\left(e_{1}\right) \oplus F_{1}^{+}\left(e_{2}\right)  \tag{18}\\
F_{1}^{++}={ }^{0}\left(f _ { 1 } ^ { 1 } \left(U_{2} e_{2} f_{0}, U_{2} e_{1} f_{0}, U_{1} e_{1} f_{2}, U_{1} e_{2} f_{2}^{2}\left(U_{1} e_{1} f_{*}, U_{1} e_{2} f_{3}^{3}\left(U_{3} e_{2} f_{3}, U_{2} e_{2} f_{1},\right.\right.\right.\right.  \tag{19}\\
\left.\left.U_{1} e_{2} f_{4}^{4}\left(U_{1} e_{1} f_{3}\right)^{4} \ldots\right)^{1}, f_{*}^{1}\left(U_{*} e_{1} f_{1}\right)^{1}\right)^{0}
\end{gather*}
$$

The structural components belong to the $e_{i}$ group of particles

$$
\begin{equation*}
E_{1}^{++}={ }^{0}\left(e_{1}^{1}\left(z_{4} p_{1} e_{2}^{2}\left(z_{4} p_{1} e_{1}\right)^{2}\right)^{1}\right)^{0} \tag{20}
\end{equation*}
$$

3.4. The first user's in operational part

The analysis of the Operational part of the algorithm is carried parallel to the analysis of the declaration and structural part [3]. Arithmetical and logical operations are inscribed into the algorithm structure

$$
\begin{equation*}
R_{1}^{+}=r_{1} e_{1}, e_{2} r_{1}, e_{2} r_{2}, e_{2} r_{1}, e_{2} r_{3}, e_{2} r_{4}, r_{0} \tag{21}
\end{equation*}
$$

Table 4
Table R

| Symbolic names | Symbolic form | Adress in $R_{\text {ext }}{ }^{++}$ |
| :---: | :--- | :---: |
| $r_{1}$ | $m_{1}:=$ to $m_{2}$ | $y_{1}$ |
| $r_{2}$ | $m_{1}:=0$ | $y_{5}$ |
| $r_{3}$ | $m_{1}:=m_{1}+m_{2}\left[m_{3}, m_{4}\right]^{*} m_{5}\left[m_{4}, m_{6}\right]$ | $y_{8}$ |
| $r_{4}$ | $m_{1}\left[m_{2}, m_{3}\right]:=m_{4}$ | $y_{11}$ |

The obtained sequence of operations should be inscribed into the structure of the algorithm, taking into consideration these statements in which we should look for the data connected with the operation types mentioned in the expression. Operation $r_{1}$, for instance, makes use of the data which occur in the declaration $k_{4}$ in position $1,3,4,5$ and in the declaration $k_{1}$ in the first position. The obtained sequence should be written as a symbolic expression $R_{1}^{++}$

$$
\begin{equation*}
R_{1}^{++}={ }^{0}\left(r_{1}^{1}\left(p_{1} e_{1} k_{1} r_{1}, p_{1} e_{2} k_{3} r_{3}, p_{1} e_{2} k_{2} r_{2}^{2}\left(p_{1} e_{2} k_{5} r_{1}\right)^{2}\right)^{1}, r_{3}^{1}\left(p_{1} e_{2} k_{4} r_{4}^{2}\left(p_{1} e_{2} k_{5} r_{0}\right)^{2}\right)^{1}\right)^{0} \tag{22}
\end{equation*}
$$

Similarly, one should expand the knowledge in the declaring, structural and operational part. The full description of the base of knowledge is in Fig. 3.


Fig. 3. Structures of the knowledge bases
Rys. 3. Obraz bazy wiedzy komputera

## 4. Enriching knowledge bases

4.1. Enriching knowledge bases in verbal description of the task

The initial knowledge basis formed on the ground of the first user's algorithm is being enriched on the basis of the new algorithms from other users. The hitherto existing knowledge basis is enriched by the elements which occur for the first time. That is because the integration of the elements in the knowledge basis depends on the phenomenon that the same knowledge element is remembered only once, no matter how many times and in how many algorithms this very element will occur. Let us consider the second user's programme allowing tabulating the function square of $[d x, d y]$ at step $d x$. The commentary to this task will be

Tabulate function square at step dx of interval $x$ to $y$

| $b_{6}$ | $\rightarrow$ | $g_{4}$ | $\rightarrow$ | Tabulate |
| :--- | :--- | :--- | :--- | :--- |
| $b_{7} b_{8} b_{9} b_{10} b_{11}$ | $\rightarrow$ | $g_{5}$ | $\rightarrow$ | function square at step $d \boldsymbol{x}$ |
| $b_{12} b_{13} b_{14} b_{15} b_{16}$ | $\rightarrow$ | $g_{6}$ | $\rightarrow$ | of interval dx and $\boldsymbol{d} \boldsymbol{y}$ |

The symbolic expressions describing group for the second user's programme has the form

$$
\begin{equation*}
B^{+}\left(g_{4}\right)={ }^{0}\left(b_{6}{ }^{1}\left(z_{4} g_{4} b_{6}\right)^{1}\right)^{0} \tag{23}
\end{equation*}
$$

$$
\begin{gather*}
B^{+}\left(g_{5}\right)={ }^{0}\left(b _ { 7 } { } ^ { 1 } \left(z _ { 1 } g _ { 5 } b _ { 8 } { } ^ { 2 } \left(z_{1} g_{5} b_{9}{ }^{3}\left(z_{1} g_{5} b_{10}{ }^{4}\left(z_{1} g_{5} b_{11}{ }^{5}\left(z_{4} g_{5} b_{7}\right)^{5} \ldots\right)^{1}\right)^{0}\right.\right.\right.  \tag{24}\\
B^{+}\left(g_{6}\right)={ }^{0}\left(b _ { 1 2 } { } ^ { 1 } \left(z _ { 1 } g _ { 6 } b _ { 1 3 } { } ^ { 2 } \left(z_{1} g_{6} b_{14}{ }^{3}\left(z_{1} g_{6} b_{15}^{4}\left(z_{1} g_{6} b_{16}^{5}\left(z_{4} g_{6} b_{12}\right)^{5} \ldots\right)^{5}\right)^{0}\right.\right.\right. \tag{25}
\end{gather*}
$$

All the symbolic expressions are united, thus forming a comprehensive expression. For second user's programme has the form

$$
\begin{gather*}
B_{2}^{++}=B^{+}\left(g_{4}\right) \oplus B^{+}\left(g_{5}\right) \oplus B^{+}\left(g_{6}\right)  \tag{26}\\
B_{2}^{+}={ }^{0}\left(b_{6}{ }^{1}\left(z_{1} g_{4} b_{6}\right)^{1}, b_{7}^{1}\left(z _ { 1 } g _ { 5 } b _ { 8 } ^ { 2 } \left(z _ { 1 } g _ { 5 } b _ { 9 } ^ { 3 } \left(z_{1} g_{5} b_{10}{ }^{4}\left(z_{4} g_{5} b_{11}{ }^{5}\left(z_{4} g_{5} b_{7}\right)^{5} \ldots\right)^{1},\right.\right.\right.\right.  \tag{27}\\
b_{12}{ }^{1}\left(z _ { 1 } g _ { 6 } b _ { 1 3 } { } ^ { 2 } \left(z_{1} g_{6} b_{14}{ }^{3}\left(z_{1} g_{6} b_{15}^{4}\left(z_{1} g_{6} b_{16}{ }^{5}\left(z_{4} g_{6} b_{12}\right)^{5} \ldots\right)^{1}\right)^{0}\right.\right.
\end{gather*}
$$

The structure of the commentary for second user's programme is a linear sequence of word groups

$$
\begin{gather*}
C_{2}^{+}={ }^{0}\left(g_{4}^{1}\left(z_{1} c_{2} g_{5}^{2}\left(z_{1} c_{2} g_{5}^{3}\left(z_{4} c_{2} g_{6}\right)^{3}\right)^{2}\right)^{1}\right)^{0}  \tag{28}\\
A_{2}^{+}={ }^{0}\left(c_{2}^{1}\left(z_{4} p_{2} c_{2}\right)^{1}\right)^{0} \tag{29}
\end{gather*}
$$

Further analysis is to put the knowledge from the commentary of the second user on the knowledge, which has already been in the base of knowledge. The result of the overlap are the expressions form

$$
\begin{align*}
& B_{\text {ext }}{ }^{++}=B_{1}^{++} \oplus B_{2}^{++}  \tag{30}\\
& C_{\text {ext }}^{++}=C_{1}^{++} \oplus C_{2}^{* *}  \tag{31}\\
& A_{\text {ext }}{ }^{++}=A_{1}^{++} \oplus A_{2}^{++} \tag{32}
\end{align*}
$$

The result of the overlap are the expression form

$$
\begin{gather*}
B_{\text {ext }}++={ }^{0}\left(b_{1}{ }^{1}\left(z_{4} g_{1} b_{1}\right)^{1}, b_{2}{ }^{1}\left(z_{1} g_{2} b_{3}{ }^{2}\left(z_{4} g_{2} b_{2}\right)^{2}\right)^{1}, b_{4}{ }^{1}\left(z_{1} g_{3} b_{2}{ }^{2}\left(z_{4} g_{3} b_{5}{ }^{3}\left(z_{4} g_{3} b_{4}\right)^{3}\right)^{2}\right)^{1},\right. \\
b_{6}{ }^{1}\left(z_{1} g_{4} b_{6}\right)^{1}, b_{7}{ }^{1}\left(z _ { 1 } g _ { 5 } b _ { 8 } { } ^ { 2 } \left(z _ { 1 } g _ { 6 } b _ { 9 } { } ^ { 3 } \left(z_{1} g_{5} b_{10}{ }^{4}\left(z_{1} g_{6} b_{11}{ }^{2}\left(z_{4} g_{5} b_{7}\right)^{5} \ldots\right)^{1},\right.\right.\right.  \tag{33}\\
b_{12}{ }^{1}\left(z _ { 1 } g _ { 6 } b _ { 1 3 } { } ^ { 2 } \left(z_{1} g_{6} b_{14}{ }^{3}\left(z_{1} g_{6} b_{15}{ }^{4}\left(z_{1} g_{6} b_{16}{ }^{5}\left(z_{4} g_{6} b_{12}\right)^{5} \ldots\right)^{1}\right)^{0}\right.\right. \\
C_{e x t}^{++}={ }^{0}\left(g_{1}{ }^{1}\left(z_{1} c_{1} g_{2}{ }^{2}\left(z_{1} c_{1} g_{3}{ }^{3}\left(z_{4} c_{1} g_{1}\right)^{3}\right)^{2}\right)^{1}, g_{4}{ }^{1}\left(z_{1} c_{2} g_{5}{ }^{2}\left(z_{1} c_{2} g_{6}{ }^{3}\left(z_{4} c_{2} g_{4}\right)^{3}\right)^{1}\right)^{0}\right.  \tag{34}\\
A_{e x t}^{++}={ }^{0}\left(c_{1}^{1}\left(z_{4} p_{1} c_{1}\right)^{1}, c_{2}{ }^{1}\left(z_{4} p_{2} c_{2}\right)^{1}\right)^{0} \tag{35}
\end{gather*}
$$

### 4.2. Enriching knowledge bases in declaration part

The declaration part from the first user's programme, which allows tabulating the function square of $[x p, y k]$ at step $d x$ has the following form
(The first form of declaration)

```
const n=20;
var wx, wy: array [l..n] of real;
i:l.. n;
x, y, xp, xk, dx, a,b, c: real;
```

(The second form of declaration)

```
const v = x;
var s, s: array [ l.. v] of real;
s:1 .. v;
s, s, s, s, s, s, s, s:real;
```

New rules:

> IF var $s, s:=\operatorname{array}[1 . . v, 1 . . v]$ of real THEN $k_{6}$
> IF $s: l . . v$ THEN $k_{7}$
> IF $s, s, s, s, s, s, s, s$ real THEN $k_{8}$

Having gained symbols $k_{i}$, the declaration components are then subjected to transformation into symbolic expression. For $k_{6}$ declaration component the expression has the form [6]

$$
\begin{equation*}
K_{6}^{+}={ }^{0}\left(l _ { 3 } { } ^ { 1 } \left(z _ { 1 } k _ { 6 } l _ { 1 4 } { } ^ { 2 } \left(z _ { 1 } k _ { 6 } l _ { 6 } ^ { 3 } \left(z _ { 1 } k _ { 6 } l _ { 1 5 } { } ^ { 4 } \left(z_{1} k_{6} l_{8}^{5}\left(z_{1} k_{6} l_{9}{ }^{6}\left(z_{4} k_{6} l_{3}\right)^{6} \ldots\right)^{0}\right.\right.\right.\right.\right. \tag{36}
\end{equation*}
$$

All the symbolic expression are united, thus forming a comprehensive expression representing a set of declaration components

$$
\begin{equation*}
K_{\text {ext }}{ }^{++}=K_{1}^{++} \oplus K_{2}^{++} \tag{37}
\end{equation*}
$$

$K_{\text {ext }}{ }^{++}={ }^{0}\left(l_{1}{ }^{1}\left(z_{1} k_{1} l_{4}{ }^{2}\left(z_{4} k_{1} l_{1}\right)^{2}\right)^{1}, l_{2}{ }^{1}\left(z_{1} k_{2} l_{5}{ }^{2}\left(z_{1} k_{2} l_{6}^{3}\left(z_{1} k_{6} l_{15}, z_{1} k_{2} l_{7}{ }^{4}\left(z_{1} k_{2} l_{8}^{5}\left(z_{1} k_{6} l_{9}\right.\right.\right.\right.\right.\right.$,
$z_{1} k_{2} l_{9}{ }^{6}\left(z_{4} k_{5} l_{13}, z_{4} k_{2} l_{2}, z_{4} k_{6} l_{3}, z_{4} k_{8} l_{16}{ }^{7}\left(z_{1} k_{8} l_{9}\right)^{7} \ldots\right)^{1}, l_{3}{ }^{1}\left(z_{1} k_{6} l_{14}, z_{1} k_{3} l_{10}{ }^{2}\left(z_{1} k_{3} l_{11}\right.\right.$,
$\left.\left.\left.z_{1} k_{4} l_{12}{ }^{3}\left(z_{4} k_{4} l_{10}\right)^{3}\right)^{2}\right)^{1}, l_{11}{ }^{1}\left(z_{4} k_{3} l_{3}, l_{13}{ }^{2}\left(z_{1} k_{5} l_{9}, z_{1} k_{7} l_{11}{ }^{3}\left(z_{4} k_{4} l_{10}\right)^{3}\right)^{2}\right)^{1}, l_{14}{ }^{1}\left(z_{1} k_{6} l_{6}\right)^{1}, l_{15}{ }^{1}\left(z_{1} k_{6} l_{9}\right)^{1}\right)^{0}$
The structure of the programme is expressed by a sequence of $V_{2}$ declaration components

$$
V_{2}=k_{1}, k_{6}, k_{7}, k_{8}, k_{0}
$$

The symbolic expression describing the structure of the declaration part has the form

$$
\begin{equation*}
V_{2}^{+}={ }^{0}\left(k _ { 1 } ^ { 1 } \left(z _ { 1 } t _ { 2 } k _ { 6 } ^ { 2 } \left(z_{1} t_{2} k_{7}^{3}\left(z_{1} t_{2} k_{8}^{4}\left(z_{4} t_{2} k_{0}\right)^{4} \ldots\right)^{0}\right.\right.\right. \tag{39}
\end{equation*}
$$

In the declaration structures there occurs only one $t$ parameter group written as

$$
\begin{gather*}
T_{2}^{+}={ }^{0}\left(t_{2}{ }^{1}\left(z_{4} p_{2} t_{2}\right)^{1}\right)^{0}  \tag{40}\\
V_{\text {ext }}^{++}=V_{1}^{++} \oplus V_{2}^{++}  \tag{41}\\
T_{\text {ext }}^{++}=T_{1}^{++} \oplus T_{2}^{* *}  \tag{42}\\
V_{\text {ext }}^{++}={ }^{0}\left(k _ { 1 } ^ { 1 } \left(z_{1} t_{2} k_{6}, z_{1} t_{1} k_{2}^{2}\left(z _ { 1 } t _ { 1 } k _ { 3 } ^ { 3 } \left(z_{1} t_{1} k_{4}^{4}\left(z_{1} t_{1} k_{5}^{5}\left(z_{4} t_{1} k_{0}\right)^{5} \ldots\right)^{1},\right.\right.\right.\right. \\
k_{6}^{1}\left(z_{1} t_{2} k_{7}^{2}\left(z_{1} t_{2} k_{8}^{3}\left(z_{4} t_{2} k_{0}\right)^{3} \ldots\right)^{0}\right.  \tag{43}\\
T_{\text {ext }}{ }^{++}={ }^{0}\left(t_{1}^{1}\left(z_{4} p_{1} t_{1}\right)^{1}, t_{2}^{1}\left(z_{4} p_{2} t_{2}\right)^{t}\right)^{0} \tag{44}
\end{gather*}
$$

4.3. Enriching knowledge bases in structural part

The structural part from the first user's programme, which allows tabulating the function square of $[x p, y k]$ at step $d x$ has the following form

```
x:=xp;
dx:=(xk-xp)/n;
for i:=l to n do
if x<xk then
begin
y:=A*x*x+B*x+C;
wx[i]:=x;
wy[i]:=y;
x:=x+dx;
end;
fori:=l to n do
writeln (wx[i]:5:2,wy[i]:5:2);
end;
```

Rules have the following form
IF begin op next.end THEN $f_{5} U_{1} U_{2} U_{3}$
IF for op do THEN $f_{6} U_{1}$
IF if op then THEN $f_{7} U_{1}$
IF begin op end THEN $f_{8} U_{0}$
Having gained symbols $f_{7}$ the structural components are the subjected to transformation into symbolic expression which for the structural component takes the following form

$$
\begin{gather*}
d_{8} d_{3} d_{9} \rightarrow f_{7} U_{1} \\
G_{7}^{+}={ }^{0}\left(d _ { 8 } { } ^ { 1 } \left(z_{1} f_{2} d_{3}{ }^{2}\left(z_{1} f_{7} d_{9}{ }^{3}\left(z_{4} f_{7} U_{1}\right)^{3} \ldots\right)^{0}\right.\right. \tag{45}
\end{gather*}
$$

Other structural components are written in an analogous way. A comprehensive symbolic expression which represents the basis of the computer knowledge in the field of structural components is obtained after having completed the covering operation

$$
\begin{gather*}
G_{2}^{++}=G_{1}^{+} \oplus G_{5}^{+} \oplus G_{6}^{+} \oplus G_{7}^{+} \oplus G_{8}^{+}  \tag{46}\\
G_{\text {ext }}{ }^{++}=G_{1}^{++} \oplus G_{2}^{++}  \tag{47}\\
G_{\text {ext }}^{++}={ }^{0}\left(d _ { 1 } ^ { 1 } \left(z_{1} f_{1} d_{5}, z_{1} f_{5} d_{3}, z_{1} f_{8} d_{3}, z_{1} f_{3} d_{3}^{2}\left(z_{1} f_{3} d_{5}, z_{1} f_{4} d_{4}, z_{1} f_{6} d_{4}, z_{1} f_{7} d_{9},\right.\right.\right. \\
z_{1} f_{8} d_{6}, z_{1} f_{5} d_{5}, z_{1} f_{2} d_{4}^{3}\left(z_{1} f_{4} d_{7}, z_{4} f_{6} U_{1}, z_{1} f_{2} d_{5}^{4}\left(z_{4} f_{2} U_{1}, z_{1} f_{3} d_{7}, z_{1} f_{5} d_{6}, z_{1} f_{1} d_{6},\right.\right. \\
\left.{ }^{5}\left(z_{4} f_{1} U_{1} U_{2}, z_{4} f_{5} U_{1} U_{2} U_{3}, z_{4} f_{8} U_{0}, z_{4} f_{3} U_{1} U_{2} U_{3}\right)^{5} \ldots\right)^{1}, d_{2}^{1}\left(z_{1} f_{4} d_{3}, z_{1} f_{2} d_{3}, z_{1} f_{6} d_{3}\right)^{1}, \\
\left.d_{7}^{1}\left(z_{4} f_{4} U_{1}, z_{1} f_{3} d_{6}\right)^{1}, d_{8}^{1}\left(z_{1} f_{7} d_{3}\right)^{1}, d_{9}^{1}\left(z_{4} f_{7} U_{1}\right)^{1}\right)^{0} \tag{48}
\end{gather*}
$$

The structural of the algorithm is presented by the symbolic expression below

$$
\begin{gather*}
F_{2}^{+}=f_{5} U_{1} U_{2} U_{4}, f_{6} U_{1}, f_{7} U_{1}, f_{8} U_{0}, f_{4} U_{1}  \tag{49}\\
e_{3}\left(f_{5}, f_{6}, f_{7}, f_{8}, f_{4}\right) \tag{50}
\end{gather*}
$$

$$
\begin{gather*}
F_{2}^{+}\left(e_{3}\right)={ }^{0}\left(f _ { 5 } ^ { 1 } \left(U _ { 1 } e _ { 3 } f _ { 6 } ^ { 2 } \left(U _ { 1 } e _ { 3 } f _ { 7 } ^ { 3 } \left(U _ { 1 } e _ { 3 } f _ { 8 } ^ { 4 } \left(U_{0} e_{3} f_{4}^{5}\left(U_{1} e_{3} f_{5}^{6}\left(U_{2} e_{3} f_{0}\right)^{6} \ldots\right)^{0}\right.\right.\right.\right.\right.  \tag{51}\\
F_{1}^{++}=F_{1}^{+}\left(e_{1}\right) \oplus F_{1}^{+}\left(e_{2}\right) \oplus F_{2}^{+}\left(e_{3}\right)  \tag{52}\\
F_{1}^{++}={ }^{0}\left(f _ { 1 } ^ { 1 } \left(U_{2} e_{2} f_{0}, U_{2} e_{1} f_{0}, U_{1} e_{1} f_{2}, U_{1} e_{2} f_{2}^{2}\left(U_{1} e_{1} f_{*}, U_{1} e_{2} f_{3}^{3}\left(U_{3} e_{2} f_{3}, U_{2} e_{2} f_{1},\right.\right.\right.\right. \\
\left.U_{1} e_{2} f_{4}^{4}\left(U_{1} e_{1} f_{3}\right)^{4} \ldots\right)^{1}, f_{5}^{1}\left(U_{2} e_{3} f_{0}, U_{1} e_{3} f_{6}^{2}\left(U_{1} e_{3} f_{7}^{3}\left(U_{1} e_{3} f_{8}^{4}\left(U_{0} e_{3} f_{4}\right)^{4} \ldots\right)^{1},\right.\right.  \tag{53}\\
\left.f_{*}^{1}\left(U_{*} e_{1} f_{1}\right)^{1}\right)^{0}
\end{gather*}
$$

The structural components belong to the $e_{i}$ group of particles

$$
\begin{gather*}
E_{2}^{++}={ }^{0}\left(e_{3}^{1}\left(z_{4} p_{2} e_{3}\right)^{1}\right)^{0}  \tag{54}\\
E_{\text {ext }}^{++}=E_{1}^{++} \oplus E_{2}^{++}  \tag{55}\\
E_{\text {ext }}^{++}={ }^{0}\left(e_{1}^{1}\left(z_{1} p_{1} e_{2}^{2}\left(z_{4} p_{1} e_{1}\right)^{1}, e_{3}^{1}\left(z_{4} p_{2} e_{3}\right)^{1}\right)^{0}\right. \tag{56}
\end{gather*}
$$

### 4.4. Enriching knowledge bases in operation part

The initial knowledge basis formed on the ground of the first user's algorithm is being enriched on the basis of the new algorithms from other users. It refers to the verbal description of the task, the declaration part and the structural as well as the operational part. The hitherto existing knowledge basis is enriched by the elements which occur for the first time. That is because the integration of the elements in the knowledge basis depends on the phenomenon that the same knowledge element is remembered only once, no matter how many times and in how many algorithms this very element will occur

$$
\begin{equation*}
R_{2}^{+}=e_{3} r_{5}, e_{3} r_{1}, e_{3} r_{6}, e_{3} r_{7}, e_{3} r_{1}, e_{3} r_{8}, r_{0} \tag{57}
\end{equation*}
$$

The obtained sequence should be written as a symbolic expression $R_{2}{ }^{++}$

$$
\begin{align*}
& R_{2}^{++}={ }^{0}\left(r_{5}^{1}\left(p_{2} e_{3} k_{6} r_{1}^{2}\left(p_{2} e_{3} k_{7} r_{8}, p_{2} e_{3} k_{7} r_{6}^{3}\left(p_{2} e_{3} k_{9} r_{1}\right)^{3}\right)^{2}\right)^{1}, r_{8}^{1}\left(p_{2} e_{3} k_{10} r_{0}\right)^{1}\right)^{0}  \tag{58}\\
& R_{e x t}^{++}=R_{1}^{++} \oplus R_{2}^{++}  \tag{59}\\
& R_{e x t}^{++}={ }^{0}\left(r_{1}^{1}\left(p_{1} e_{1} k_{1}^{\prime} r_{1}, p_{1} e_{2} k_{3}^{\prime} r_{3}, p_{2} e_{3} k_{7}^{\prime} r_{8}, p_{2} e_{3} k_{7}^{\prime} r_{6}, p_{1} e_{2} k_{2}^{\prime} r_{2}^{2}\left(p_{1} e_{2} k_{5}^{\prime} r_{1}\right)^{2}\right)^{1},\right. \\
&\left.r_{3}^{1}\left(p_{1} e_{2} k_{4}^{\prime} r_{4}^{2}\left(p_{1} e_{2} k_{5}^{\prime} r_{0}\right)^{2}\right)^{1}\right)^{0}, r_{5}^{1}\left(p_{2} e_{3} k_{6}^{\prime} r_{1}\right)^{1}, r_{6}^{1}\left(p_{2} e_{3} k_{8}^{\prime} r_{7}^{2}\left(p_{2} e_{3} k_{7}^{\prime} r_{1}\right)^{2}\right)^{1}, \\
&\left.r_{8}^{1}\left(p_{2} e_{3} k_{10}^{\prime} r_{0}\right)^{1}\right)^{0} \tag{60}
\end{align*}
$$

## 5. Realization of the base of knowledge

For a computer to have the ability of automatic programming it is necessary to possess adequate knowledge. This knowledge is contained in the user's programmes written in the language of higher level. The aim of the present paper is to determine acceptable user's programmes transformation into a symbolic form which constitutes an element of the knowledge basis. The knowledge basis of such a computer should be expressed in an
adequate symbolic form, thus allowing maximum knowledge integration in the computer memory. To realize knowledge integration it is assumed that one and the very one element of knowledge should be represented once only, no matter in how many programmes and how many times it will appear. In the paper the concept of acquiring knowledge contained in statements and in operation code sequences has been formalized, as well as its representation in the knowledge basis [1, 2].

Special attention was paid to the declaration part. It is worth mentioning that the formalism introduced in the paper, as well as the concept of the computer knowledge basis organization forms the ground for further research in the field of automatic programming. The concept of the knowledge basis organization presented in the paper should be, in the future, developed in the direction of forming new algorithms from the knowledge elements belonging to different tasks. A different sequence of the same components may an algorithm capable of solving a new problem. To verify the elaborated algorithms a synthesis was carried out of the user's programme on the basis of the user's instruction expressed in natural language Fig. 4.


Fig. 4. Knowledge in the field of understanding the natural language Rys. 4. Algorytm syntezy programów użytkownika

When the knowledge basis is of the software kind the knowledge elements are stored in memory cells and the process of concluding is complex and time consuming. It is very different from the human way of drawing conclusions. While thinking or regenerating
the acquired knowledge a human being transforms a semantic picture which can not their appears the necessity to realize a part of the basis through hardware [4].

Hardware is a permanent structure which can be based on e.g. transferors. A chosen configuration of transferors may represent the semantic picture of a sentence. The software part of the knowledge bases is therefore used to perform a syntactic analysis. The result of this analysis, that is the symbolic expressions, are the transformed to the hardware part. The knowledge representation can be shown as a graph, which univocally has a repercussion on the logical structure of the hardware.

As the knowledge is acquired and accumulated the semantic memory will absorb newer and newer semantic pictures. If at the input of such a hardware we ask a question in an adequate symbolic form, then it should activate a certain part of its structure which would generate an answer in an accepted knowledge representation and which would be then transformed into a sentence in natural language.

A hardware which is able to adjust its logical structure to a symbolic representation of the knowledge bases is called an evolvable hardware. Hardware is steady structure. The semantic picture may be represented by appropriate scheme of this flip-flop. The symbolic expressions are this form, of representation of the knowledge, which can be shown as a graph. Such form definitely depicts the logical structure of the hardware [7].

Every user's program could be believed to be a specific kind of semantic picture. The more knowledge the semantic memory will take from the user's program the semantic pictures it will contain. Hardware, which can match its logical structure to the symbolic representation of the base of knowledge is called the evolvable hardware.

## 6. Conclusions

In this paper an approach to automatic synthesis of programs from pieces of knowledge acquired to the knowledge base from another different programs has been introduced. The concept of the knowledge bases organization presented in the paper should be, in the future, developed in the direction of forming new algorithms from take knowledge elements belonging to different tasks. A different sequence of the same components may an algorithm capable of solving a new problem.

## References

[1] Kazimierczak J., Knowledge Representation on the Level of Natural Language for Purposes od Automatic Programming, Proceeclings of the Inter., Conference, Rockville, Maryland, USA., KSI Press, Skokie 1995.
[2] Kazimierczak J., Zastosowanie ewolucyjnego hardware'u do reprezentacji wiedzy wyrażonej w postaci zdań jezyka naturalnego, Proc. of $4^{\text {th }}$ Konferencja nt. „Inżynieria wiedzy i systemy ekspertowe", Vol. 1, Wrocław 2000, 235-243.
[3] Plichta S., Synteza bazy wiedzy komputera w zakresie czę́ci operacyjnej i deklarujacej dla potrzeb automatyzacji programowania, praca doktorska, Politechnika Wrocławska, Wrocław 1991.
[4] Plichta S., Evolvable Hardware for the Needs of Programming Automatisation, VI International Conference on Artifical Inteligence AI-19’ 2004, Siedlce 2004.
[5] Plichta S., Synthesis of computer knowledge basis for the needs of programming automatisation, 10TH International Conference on System-Modelling-Control, Zakopane 2001.
[6] Plichta S., Formal model evolvable hardware to representation of knowledge for automatic programming, AI MECH 2002, Methods of Artiflcial Intelligence in Mechanics, Gliwice 2002.
[7] Xin Yao, Tetsuya Higuchi, Promises and Chailanges of Evolve Hardware, Proc. of the $1^{\text {st }}$ Intern. Conf. on Evolvable Systems: from Biology to Hardvare, Tskuba, Japan, Springer Press, Tokyo 1997, 55-80.


[^0]:    * Dr inż. Stanisława Plichta, Instytut Modelowania Komputerowego, Wydział Fizyki, Matematyki i Informatyki Stosowanej, Politechnika Krakowska; Katedra Informatyki, Wydział Nauk Ekonomicznych, Wyższa Szkoła Umiejętności im. Stanisława Staszica w Kielcach.

