An Intelligent Iconic System to generate and to interpret Visual Languages

C. Crimi, A. Guercio, G. Tortora, M. Tucci

Dipartimento di Informatica ed Applicazioni
University of Salerno, Italy

1. Introduction

The use of visual aids in man-machine interfaces is spreading rapidly thanks to the convergence of new low cost technology in the fields of graphics, image processing, video and microelectronics and by the growing interest in multimedia communications.

Thus, the demand has increased for user-friendly iconic programming systems for both non-expert users and professionals. Iconic systems have been proposed [1, 4, 6, 7, 8, 11, 12] as a valuable interface and since icons directly represent concepts [4, 10] and thus all functions of such concepts are automatically represented by icons. Furthermore, as shown in [9], iconic systems for specialized applications are more immediate both for learning and use. Hence, iconic programs are used to obtain higher man-machine interfaces. Secondly, the ability of icons images have, to be intuitively understood by most users.

As Norman and Tashler [10] pointed out, systems should be specified by a conceptual model which has to describe the system in terms of the mental model of an ideal user and has to preserve the whole man-machine interface. As a result, the image of the system the user wants will be intelligible, consistent and cohesive to the conceptual model. Moreover, the system conceptual model should be explained to the user according to the main principles of Learnability, Functionality and Usability.

To maximize the interaction of the user's concepts and the system's models, an iconic system has been proposed utilizing the user's concepts to define the appropriate visual language. Efficiency and consistency are thus achieved, together with learnability, functionality and usability, so that icons will directly reflect the user interpretation of a concept.

Using the SIL-Icon compiler developed in [4, 11, 12], whereby a general-purpose iconic system is specialized by expert provision of domain-specific Grammar and Semantic to define the individual visual language, a Learning Interface Module (LIM, in the sequel) has been designed allowing the non-expert user to utilize the system and define his own visual language providing visually accessible knowledge from which the LIM autonomously constructs the domain-specific aspects of the language.

A brief description of the SIL-Icon compiler is presented in section 2. After giving an overview of the whole learning iconic system, all the principal concepts and modules are detailed in Sections 3 - 8.

2. The SIL-ICON Compiler v.3

Let us recall some definitions introduced in [12]. An iconic system is a structured set of related icons that are the dual representation of an object. An icon is denoted by \((X_{i}, E_{i})\) where \(X_{i}\) is the logical part (the meaning) and \(E_{i}\) is the physical part (the image). A visual sentence is a spatial arrangement of icons from an iconic system obtained combining icons from the iconic system by means of horizontal concatenation (\(\searrow\)), vertical concatenation (\(\downarrow\)) and spatial overlapping (\(\uparrow\)).

The elementary icons are those to be primitive icons of the system and they are partitioned in elementary object icons and process icons. Elementary object icons identify objects while process icons express computations. Composite icons are obtained by spatial arrangement of elementary object icons.

The SIL-Icon Compiler v.3 is a software system for the specification, interpretation, prototyping and generation of iconic-oriented systems. The system diagram includes the following parts:

\[\text{Pattern Analysis} \quad \text{Syntax Analysis} \quad \text{Construction of the logical part} \quad \text{Meaning of } S \]
3. The learning iconic system

The iconic system presented in this section is an intelligent man-machine interface. In Fig. 2, the diagram of the iconic system is shown.

![Diagram of the iconic system](image)

The system embodies an improved version of the Pattern Analyzer (PA), the Learning Interface Module (LIM) and the Icon Interpreter described in the previous section. The Pattern Analyzer carries out the structural analysis of the visual sentences and sends its output either to the Icon Interpreter or to the Learning Interface Module. LIM constructs the core of the learning part of the system since it supports the user in the definition of his own visual language. On this purpose, a sample set of visual sentences S from the desired language is input to LIM, which infers a context-free language L and the grammar G, with S\(\in\)L(G), thanks to the detection of the structural features of the elements of S. Moreover, the Icon Dictionary (ID) and the Knowledge Base (KB) connected to the inferred language are created. It is worth noting that the Learning Module specializes the iconic system on the user's suggestions, therefore the system is strictly modelled on the user's mental concepts [10].

The system can operate in two different states:

a) the generation of an iconic language (system initialization phase);

b) the use of the language created in state a) (execution phase).

The system is configured during the initialization phase, in which the visual sentences are syntactically analyzed by PA, transformed into trees, and sent to LIM, which creates G, ID and KB thanks to the user interaction. When the initialization phase is over, the system is ready to input visual programs and to execute them (case b) by consulting the previously created G, ID and KB. The Pattern Analyzer and the Icon Interpreter are the only active modules during the execution phase.

4. Learning Icon System specification

The learning iconic system proposed is a general purpose system and can be used as a natural interface between a target language of the underlying system and the user's language. In this section, the target language for the correct specialization of the iconic system is described.

The information needed to specify the iconic system are:

a) the set of data types of the objects, \(\{I\} = \{\ldots, i, \ldots\}\), is provided and for each compound data type \(l\), the elementary data types composing \(l\) are known.

b) the set of functions, \(\{f\} = \{\ldots, f, \ldots\}\), is provided and for each function \(f\), the number of operands, the data type of each operand and the data type of the resulting objects (if there exist) are known.

Information about the functions of the target language, the data types of the objects, together with the possible spatial arrangements allowed for them, are collected in two tables:

1) the table of the objects;

2) the table of the functions.

**Example 4.1**

Let us give the following table of objects. Let image be the set of all the images associated to object icons.

<table>
<thead>
<tr>
<th>data type</th>
<th>associated parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>images</td>
</tr>
<tr>
<td>line</td>
<td>images</td>
</tr>
<tr>
<td>marked line</td>
<td>images</td>
</tr>
<tr>
<td>file</td>
<td>images</td>
</tr>
<tr>
<td>string</td>
<td>(horizontal composition of characters) (vertical composition of characters)</td>
</tr>
<tr>
<td>block</td>
<td>(horizontal composition of lines) (vertical composition of lines)</td>
</tr>
<tr>
<td>left marked block</td>
<td>(horizontal composition of marked line and block) (vertical composition of marked line and block)</td>
</tr>
</tbody>
</table>

The table \(T\) establishes the set of all the elementary and complex objects which can be obtained by the set of data types of the target language. The table contains information about the syntactic categories of the macro general grammar \(G\), obtainable from \(T\). The language \(L(G)\), inferred by the system, has to be necessarily contained in \(L(G)\). It follows that the user cannot define any object the target language does not deal with.

For each function the table of the functions will contain information about the number of the operands, the relationships between the operands and the data types of the objects obtained after the application of the function itself to the chosen operands. Note that there are several functions which can be considered conceptually similar to others. For example, in the command "copy a string \(s\) into a string \(s'\)" the function copy is conceptually similar to the function copy of the command "copy a block \(B\) into a block \(B'\)" or "copy a block \(B_1\) into a block \(B_2\)". Therefore, we suggest to include the copy functions applied to the previous distinct operands in a unique class. We assume that the different functions applied to the same arguments cannot appear in the same class. When all the functions are collected in their respective classes, a measure of the conceptual similarity between the classes themselves can be provided.

The measure is expressed by the function \(\text{cs} : \mathcal{C} \times \mathcal{C} \to [0,1]\), where \(\mathcal{C}\) is the set of classes of functions. It is possible to define a table which, for each class \(C\in\mathcal{C}\), contains:

- a) the name of the class;
- b) for each function \(f\in C\), the number of arguments of \(f\); the data type of the arguments and finally the data type of the resulting objects, if they exist;
- c) the conceptual similarity measure with respect to the remaining classes.

**Example 5.2**

Let \(F\) be the set of functions:

- \(\text{display file(file, screen)}\), \(\text{display file(file, screen)}\)
- \(\text{display block(block, screen)}\), \(\text{display block(block, screen)}\)
- \(\text{insert string(string, screen)}\), \(\text{insert string(string, screen)}\)

145
The system. During information about the srmc~waJ analysis of step (a) is
Given a pattern, the precedence
of each visual sentence, ve
initialization phase of
as input.

The Pattern Analyzer

The input provided to the Icon system by the user has to be consistent with the target
language of the system. During the initialization phase of the system, the user is required to provide:
1) a sample set S of visual sentences, constructed using the set of images available in the system;
2) for each visual sentence, ve S, the object icons of the process icons associated with each
distinct image of ve, together with the number of arguments and their spatial arrangement;
3) the semantics of each visual sentence ve S.
We observe that in (2) when a process icon is associated with an image of ve, the number of
arguments and their spatial arrangements have to be defined. This information, together with
the information about the object icons, are collected into the Icon Dictionary.

5. The Pattern Analyzer

The Pattern Analyzer of the SIL-Icon Compiler is in charge of:

a) transforming the user's visual sentences from their initial representation into an equivalent
representation suitable for LIM and the Icon Interpreter
b) transforming the visual sentences used by LIM into their original spatial representation.
The structural analysis of step (a) is based upon the methodology of the precedence grammar [2].
Given a pattern, the precedence grammar splits the main frame (the pattern) into one or more
subframes according to some division rules.

Example 5.1 Let U be the following visual sentence

U =

The parse tree of U is

F =

In step (b) the parse tree of each U is visited, the information of the internal nodes of the parse tree,
stored in ID, allow the system to reconstruct the spatial disposition of the primitive icons which
constitute U. This is necessary, since the system has to colloquize with the user using visual
sentences in their spatial representation.

6. The Learning Interface Module

The Learning Interface Module constitutes the core of the whole system. It allows the user to
specialize the icon system on the basis of his own needs. Given a set of visual sentences S, PA
analyzes them and produces a set of trees T, the set of the pattern trees of the visual sentences in S.
T is input to LIM which, thanks to the process of syntactic inference and semantic synthesis,
generates G and KB and updates ID. During the creation of the language, the system asks for the
user's intervention, if necessary.

The module LIM, as shown in fig. 5.

fig. 4

The input provided to the icon system by the user has to be consistent with the target language of
the system. During the initialization phase of the system, the user is required to provide:
1) a sample set S of visual sentences, constructed using the set of images available in the system;
2) for each visual sentence, ve S, the object icons of the process icons associated with each
distinct image of ve, together with the number of arguments and their spatial arrangement;
3) the semantics of each visual sentence ve S.

We observe that in (2) when a process icon is associated with an image of ve, the number of
arguments and their spatial arrangements have to be defined. This information, together with
the information about the object icons, are collected into the Icon Dictionary.

5. The Pattern Analyzer

The Pattern Analyzer of the SIL-Icon Compiler is in charge of:

a) transforming the user's visual sentences from their initial representation into an equivalent
representation suitable for LIM and the Icon Interpreter
b) transforming the visual sentences used by LIM into their original spatial representation.
The structural analysis of step (a) is based upon the methodology of the precedence grammar [2].
Given a pattern, the precedence grammar splits the main frame (the pattern) into one or more
subframes according to some division rules.

Example 5.1 Let U be the following visual sentence

U =

The parse tree of U is

F =

In step (b) the parse tree of each U is visited, the information of the internal nodes of the parse tree,
stored in ID, allow the system to reconstruct the spatial disposition of the primitive icons which
constitute U. This is necessary, since the system has to colloquize with the user using visual
sentences in their spatial representation.

6. The Learning Interface Module

The Learning Interface Module constitutes the core of the whole system. It allows the user to
specialize the icon system on the basis of his own needs. Given a set of visual sentences S, PA
analyzes them and produces a set of trees T, the set of the pattern trees of the visual sentences in S.
T is input to LIM which, thanks to the process of syntactic inference and semantic synthesis,
generates G and KB and updates ID. During the creation of the language, the system asks for the
user's intervention, if necessary.

The module LIM, as shown in fig. 5.

fig. 5

The input provided to the icon system by the user has to be consistent with the target language of
the system. During the initialization phase of the system, the user is required to provide:
1) a sample set S of visual sentences, constructed using the set of images available in the system;
2) for each visual sentence, ve S, the object icons of the process icons associated with each
distinct image of ve, together with the number of arguments and their spatial arrangement;
3) the semantics of each visual sentence ve S.

We observe that in (2) when a process icon is associated with an image of ve, the number of
arguments and their spatial arrangements have to be defined. This information, together with
the information about the object icons, are collected into the Icon Dictionary.

5. The Pattern Analyzer

The Pattern Analyzer of the SIL-Icon Compiler is in charge of:

a) transforming the user's visual sentences from their initial representation into an equivalent
representation suitable for LIM and the Icon Interpreter
b) transforming the visual sentences used by LIM into their original spatial representation.
The structural analysis of step (a) is based upon the methodology of the precedence grammar [2].
Given a pattern, the precedence grammar splits the main frame (the pattern) into one or more
subframes according to some division rules.

Example 5.1 Let U be the following visual sentence

U =

The parse tree of U is

F =

In step (b) the parse tree of each U is visited, the information of the internal nodes of the parse tree,
stored in ID, allow the system to reconstruct the spatial disposition of the primitive icons which
constitute U. This is necessary, since the system has to colloquize with the user using visual
sentences in their spatial representation.

6. The Learning Interface Module

The Learning Interface Module constitutes the core of the whole system. It allows the user to
specialize the icon system on the basis of his own needs. Given a set of visual sentences S, PA
analyzes them and produces a set of trees T, the set of the pattern trees of the visual sentences in S.
T is input to LIM which, thanks to the process of syntactic inference and semantic synthesis,
generates G and KB and updates ID. During the creation of the language, the system asks for the
user's intervention, if necessary.

The module LIM, as shown in fig. 5.

fig. 5

The input provided to the icon system by the user has to be consistent with the target language of
the system. During the initialization phase of the system, the user is required to provide:
1) a sample set S of visual sentences, constructed using the set of images available in the system;
2) for each visual sentence, ve S, the object icons of the process icons associated with each
distinct image of ve, together with the number of arguments and their spatial arrangement;
3) the semantics of each visual sentence ve S.

We observe that in (2) when a process icon is associated with an image of ve, the number of
arguments and their spatial arrangements have to be defined. This information, together with
the information about the object icons, are collected into the Icon Dictionary.

5. The Pattern Analyzer

The Pattern Analyzer of the SIL-Icon Compiler is in charge of:

a) transforming the user's visual sentences from their initial representation into an equivalent
representation suitable for LIM and the Icon Interpreter
b) transforming the visual sentences used by LIM into their original spatial representation.
The structural analysis of step (a) is based upon the methodology of the precedence grammar [2].
Given a pattern, the precedence grammar splits the main frame (the pattern) into one or more
subframes according to some division rules.

Example 5.1 Let U be the following visual sentence

U =

The parse tree of U is

F =

In step (b) the parse tree of each U is visited, the information of the internal nodes of the parse tree,
stored in ID, allow the system to reconstruct the spatial disposition of the primitive icons which
constitute U. This is necessary, since the system has to colloquize with the user using visual
sentences in their spatial representation.

6. The Learning Interface Module

The Learning Interface Module constitutes the core of the whole system. It allows the user to
specialize the icon system on the basis of his own needs. Given a set of visual sentences S, PA
analyzes them and produces a set of trees T, the set of the pattern trees of the visual sentences in S.
T is input to LIM which, thanks to the process of syntactic inference and semantic synthesis,
generates G and KB and updates ID. During the creation of the language, the system asks for the
user's intervention, if necessary.

The module LIM, as shown in fig. 5.

fig. 5

The input provided to the icon system by the user has to be consistent with the target language of
the system. During the initialization phase of the system, the user is required to provide:
1) a sample set S of visual sentences, constructed using the set of images available in the system;
2) for each visual sentence, ve S, the object icons of the process icons associated with each
distinct image of ve, together with the number of arguments and their spatial arrangement;
3) the semantics of each visual sentence ve S.

We observe that in (2) when a process icon is associated with an image of ve, the number of
arguments and their spatial arrangements have to be defined. This information, together with
the information about the object icons, are collected into the Icon Dictionary.

5. The Pattern Analyzer

The Pattern Analyzer of the SIL-Icon Compiler is in charge of:

a) transforming the user's visual sentences from their initial representation into an equivalent
representation suitable for LIM and the Icon Interpreter
b) transforming the visual sentences used by LIM into their original spatial representation.
The structural analysis of step (a) is based upon the methodology of the precedence grammar [2].
Given a pattern, the precedence grammar splits the main frame (the pattern) into one or more
subframes according to some division rules.

Example 5.1 Let U be the following visual sentence

U =

The parse tree of U is

F =

In step (b) the parse tree of each U is visited, the information of the internal nodes of the parse tree,
stored in ID, allow the system to reconstruct the spatial disposition of the primitive icons which
constitute U. This is necessary, since the system has to colloquize with the user using visual
sentences in their spatial representation.
EVENT_NAME(proc_name, obj_name1, obj_name2) with arguments obj_name1 and obj_name2.

5) RESULT_REF (proc_name, obj_name1, obj_name2)
   it returns a reference to the object representing the result.

The following two tables take into account cases in which object icons are composed (composite object icons).

6) RELATION (obj_name1, obj_name2, op)
   it returns the conceptual relation existing between obj_name1 and obj_name2 when they are spatially combined by means of the operator op.

7) RESULT_OBJECT (obj_name1, obj_name2, op)
   it returns the name of the object resulting by the spatial combination (via op) of obj_name1 and obj_name2.

In the previous tables, obj_name1 and obj_name2 refer to the logical part of the icons. Besides domain-specific relations, the tables can contain also these special relations:

- <new>
  this relation means that the combination obj_name1 op obj_name2 gives rise to a new object, different from obj_name1 and obj_name2.

- <null>
  this relation means that the combination obj_name1 op obj_name2 generates an object which has the same logical part of obj_name1 or obj_name2. In other words, it points out that no additional information are derived from such combination.

- <obj> (resp. <obj>)
  this relation means that the combination obj_name1 proc_name, obj_name2 generates an object which has the same reference of the icon whose name is obj_name1 (resp. obj_name2).

For simplicity, in the following the tables will be depicted as in [9], and only the most interesting information at each time will be shown.

During SEM’s activation, two phases can be detected. In the first one, KB is initialized and ID is updated (1); in the second one, KB is updated (2).

During the construction of KB, event name fields will be filled in by class names. Since a relation describes the role of a operand of a given function, the relation name will be easily derived consulting the table of Functions of the target language description.

In phase (1) SEM visits each tree t in S in a bottom up way. Thanks to the user’s interaction and to the tables of objects and functions, it provides a semantic value for t. In phase (2), SEM analyzes the equivalence classes produced by SEM, searches for the new inferred trees and provides the semantic values for them, if possible. When more than one meaning can be assigned to a tree, the user’s intervention is required to select one of the meanings among the possible ones. Then, SEM updates and enlarges KB on the base of the acquired knowledge.

7.1 Phase (1)

As an example of initialization of KB, let U (example 5.1) be the first visual sentence introduced in the system and P be the pattern tree produced by PA. SEM searches for the current assignment of the semantic value to the following subterms:

- Line
- M-Line
- R.M.Block
- Block
- Print

Let the table of objects and the table of functions be those of the examples 4.1 and 4.2, respectively. SEM starts by considering the subterm a. Since it is composed by an image of an elementary object icon, SEM searches in the list of the associated patterns of the object table for the associated data type value. The list of the possible data types is (character, marked_character, line, marked_line, printer, file). If “line” is chosen by the user, then the system associates the symbol

with the data type “line” and updates the associated pattern of “line” with the set

Since “line” is an elementary object, the elementary object icon

is added to the Icon Dictionary.

Since the root of tree (b) is a spatial operator “A”, the table of objects has to be consulted. In the table of objects, “block” is the only data type associated with the vertical composition of “lines”. Therefore (b) assumes the meaning represented by the data type “block”. Since “block” once a spatial operator in its definition, as easy in tables of KB which defines the vertical arrangement of lines is created. The relation of the new entry is set up to “new” since the data type produced, “block”, is different from the data types of the operands, “lines”, of the spatial operator “A”.

For composite icons, like

process

the search is carried out on the table of functions. The class, which the function having obj1 and obj2 as arguments belongs to, is one of the possible candidates for the meaning of the icon itself. On the base of these considerations, the following tables are generated from the previous visual sentence.

<table>
<thead>
<tr>
<th>Line</th>
<th>M.Line</th>
<th>R.M.Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>Block</td>
<td>R.M.Block</td>
</tr>
<tr>
<td>Line</td>
<td>Block</td>
<td>R.M.Block</td>
</tr>
</tbody>
</table>

If the following visual sentence is the new input,

its pattern tree,

is analyzed. The table on the right in fig. 6 is updated producing the table on the right in fig. 7; the table on the left (fig. 7) is generated.

7.2 Phase (2)

Let us observe now what happens when SEM attempts to assign a meaning to a visual sentence produced by an inference. Notice that in phase (2) the input of SEM is the set of equivalence classes produced by SEM. Each equivalence class contains trees which can be considered similar in some way. Let us suppose that the subterm t of the tree has been found to be equivalent to the subterms a. The substitution of a with t produces a new visual sentence s’ (fig. 8).

...
Let the resulting objects of (res obj(u, v, w) is indicated by res obj(0, res obj(u), res obj(v)), res obj(w)) and opposite A (resp. C, D). Let us suppose that the meaning of res obj(Pu(C, C)) has been found. If the entry is the table of the process Pu for Pu(A, C) is defined for each A such that there exists a subtree t with res obj(t) = A and if t, then res obj(Pu(B, C)) does not necessarily have the same meaning as any res obj(Pu(res obj(t), C)) for all the trees t such that (t1: if res obj(Pu(B, C)) = res obj(res obj(t), C)) for some t, with t1, then the entry Pu(2, res obj(Pu(B, C))) is the table of process Pu is equal to the entry Pu(res obj(res obj(t), C)). Otherwise, the entry Pu(2, res obj(res obj(t), C)) has to be calculated.

This means that, even if the meaning of the tree obtained substituting t with u is known, the meaning of all the trees obtained substituting a tree in [u] with [u] in all the visual sentences containing k is not necessarily known yet. It is worth noting that each tree always represents a unique logical object, but one logical object may be represented by more than one tree. As an example, the trees t1 and t2

represent the same logical object, "string", even though t1 represents the physical object "string of two characters" and t2 the physical object "string of three characters".

In the following, the logical object associated to a tree will be denoted by C and the set of objects of each i, i by [i]. A set of visual sentences is input to the algorithm SEMANTIC SYNTHESIS, which implements the phase (2).

Algorithm SEMANTIC SYNTHESIS
1) For each object C, belonging to [i], do

begin

2) if C [i] then no operation is necessary

3) if C [i] then its values of visual sentences containing k are determined

else

4) if there is no C [i] such that C [i] is a row index of TAV then go step 5) to analyze TAV's comment

else

begin

5) if C [i] is not a row index in TAV then add a new row in TAV and assign it to C [i];

Col_index_set = [C] where C is a column index of TAV such that:

TAV(C) is defined for some [i] with [i] and for each Ce Col_index_set such that TAV(C) [i] is neither defined nor examined yet do

begin

look for the set of events associatable to TAV(C) according to the algorithm Semantic;

if C is a singleton then En-1 = E (b, E is the event field of TAV(C) if

end

= En-1 (i) selected by the user;

if not then (i) associatable to TAV(C) has been determined)

begin

define fields of TAV(C) with respect to E, according to information in the table of functions:

- res obj(TAV(C)) = res obj(TAV(C)) for any C [i] (i) then repeat step 3) substituting C [i] and [i] by C [i] and the set of remaining objects of TAV(C) where C [i] does not

end

end

if there is any C [i] such that C [i] is a column index in TAV then execute step 5) reversing the roles of columns and rows

else

7) for each object C [i] execute step 2) reversing the roles of i and u.

end SEMANTIC SYNTHESIS
Let the conceptual similarity threshold parameter $k$ between the classes of functions be set to 0.7. $S^*$ is input to SIM which discovers a relation between the subgraphs of fig. 12.

![fig. 12](image)

The system substitutes the occurrences of $x$ with $x$ in the tables of KB. SIM considers the table for $x$ and attempts to fill in the appropriate entries corresponding to the value File as new index. The functions having File and Sel block as arguments, in the class of function "Insert", corresponding to the new index equal to Block, is searched unsuccessfully. Thus, no meaning is found and the entry $n$(File, Sel block) is not filled in. The class "Output" is visited in order to assign a value to $n$(File, Video, err). Since the Display file command is the only one accepting as arguments File and Video, err, and since "Block" in the only element of OBJ[i], then

![Diagram](image)

No meaning can be assigned to the entry $n$(File, Block), therefore it is left empty. Since Block and File are in relation $R$, a new column is added having File as its index. In order to fill in the entry $n$(Block, File), SIM searches in the class of functions "Append" and the function "Append_file" is founded. Then

![Diagram](image)

Since "Append_file" produces File as result, which is different from any resulting object belonging to OBJ[i], then the entries of the table having File as a new index are analyzed without causing any entry to be filled. Finally the entry $n$(Block, Printer) is considered. SIM looks for a function in the class of functions "Output", since it is the one associated to the entry $n$(File, Printer), and the "Print_block" command is found. Then

![Diagram](image)

The final table is

<table>
<thead>
<tr>
<th>OBJ</th>
<th>Sel Block</th>
<th>Printer</th>
<th>Video file</th>
<th>Block</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ</td>
<td>Place</td>
<td>Place</td>
<td>Place</td>
<td>Place</td>
<td>Place</td>
</tr>
</tbody>
</table>

The set of visual sentences inferred by SIM and their corresponding meanings assigned by SIM are shown in the following:

![Diagram](image)

8. Interactions between SIM and SEM

As already stated, SIM's behavior during phase (2) is dependent on that of SEM. For clarity, the necessary interactions between the two modules to specify the system are elaborated in more details.

SIM seeks similarities between subtexts belonging to $S^*$. When similarities between two subtexts and a threshold is discovered and the relation $S'$ is inferred, SIM sends it to SEM and waits for a response. SEM attempts to associate a meaning to the visual sentences inferred by the new relation taking into consideration both the relation between $x$ and its current equivalence classes.

SEM can always generate the meaning of all the new sentences inferred; often, in fact, a function with appropriate operands which should allow semantic values to be assigned to the entries in visual tables does not exist. An exceptional case could also happen: occasional semantic meaning cannot be generated for any of the inferred sentences.

In these cases the heuristic nature of the inference algorithms excludes the possibility of semantic control of the inferred relations.

This creates the necessity of a control within SEM over the many inferences produced by SIM. Otherwise, an uncontrolled language may be created allowing too many sentences of low semantic meaning to be acceptable to the syntax of the language in the exceptional case described above. However such a case might allow useful inferences to be generated and so should not be ignored. Therefore a practical solution might be to provide SEM with statistical control permitting only a fixed percentage of failures in the attempts to attribute values to the entries in tables to be developed from $S^*$. In the case where there is a higher percentage of failures the relation $S'$ will be automatically rejected. The failure threshold value ($k$) thus assumes a very important role in the tuning of the system. For example if $k=0$ then all entries aimed must have value for $S'$ in the acceptable judgment to $S'$; or if $k=1$ then any relation is valid regardless of the number of the remaining sentences generated.

From a computational viewpoint such a control would be cumbersome because of the statistical valuations of failures but because of the controls necessary within SIM to avoid infinite repetitions of the same rejected $S'$.

The SIL-icon Compiler is being developed by the University of Pittsburgh and the University of Salerno.

References


149