VLSS

Journal of
Visual Languages and
Sentient Systems

Volume 4, December, 2018

Journal of Visual Languages and Sentient Systems

An Open Access Journal published by

KSI Research Inc.

156 Park Square Lane, Pittsburgh, PA 15238 USA

DOI: 10.18293/VLSS2018

Proceedings preparation, editing and printing are sponsored by KSI Research Inc.

Journal of Visual Languages and Sentient Systems

Volume 4, 2018

Table of Contents

INFORMATION FOR AUTHORS

The Journal of Visual Languages and Sentient Systems (VLSS) is intended to be a forum for researchers, practitioners and developers to exchange ideas and research results, for the advancement of visual languages and sentient multimedia systems. Sentient systems are distributed systems capable of actively interacting with the environment by gathering, processing, interpreting, storing and retrieving multimedia information originated from sensors, robots, actuators, websites and other information sources. In order for sentient systems to function efficiently and effectively, visual languages may play an important role.

VLSS publishes research papers, state-of-the-art surveys, review articles, in the areas of visual languages, sentient multimedia systems, distributed multimedia systems, sensor networks, multimedia interfaces, visual communication, multi-media communications, cognitive aspects of sensor-based systems, and parallel/distributed/neural computing & representations for multimedia information processing. Papers are also welcome to report on actual use, experience, transferred technologies in sentient multimedia systems and applications. Timely research notes, viewpoint articles, book reviews and tool reviews, not to exceed three pages, can also be submitted to VLSS.

Manuscripts shall be submitted electronically to VLSS. Original papers only will be considered. Manuscripts should follow the double-column format and be submitted in the form of a pdf file. Page 1 should contain the article title, author(s), and affiliation(s); the name and complete mailing address of the person to whom correspondence should be sent, and a short abstract (100-150 words). Any footnotes to the title (indicated by *, +, etc.) should be placed at the bottom of page 1.

Manuscripts are accepted for review with the understanding that the same work has not been and will not be nor is presently submitted elsewhere, and that its submission for publication has been approved by all of the authors and by the institution where the work was carried out; further, that any person cited as a course of personal communications has approved such citation. Written authorization may be required at the Editor's discretion. Articles and any other material published in VLSS represent the opinions of the author(s) and should not be construed to reflect the opinions of the Editor(s) and the Publisher.

Paper submission should be through: vlss@ksiresearch.org

For further information contact: vlss@ksiresearch.org

A Visual Debugging Aid Based on Discriminative Graph Mining

Jennifer L. Leopold[1], Nathan W. Eloe[2], Jeff Gould[1], and Eric Willard[1]

[1]Missouri University of Science & Technology
Department of Computer Science
Rolla, MO, USA

[2]Northwest Missouri State University
School of Computer Science and
Information Systems
Maryville, MO, USA

leopoldj@mst.edu, nathane@nwmissouri.edu, jg7f9@mst.edu, emwwwc@mst.edu

Abstract—Why doesn't my code work? **Instructors for introductory programming courses frequently are asked that question. Often students understand the problem they are trying to solve well enough to specify a variety of input and output scenarios. However, they lack the ability to identify where the bug is occurring in their code. Mastering the use of a full-feature debugger can be difficult at this stage in their computer science education. But simply providing a hint as to where the problem lies may be sufficient to guide the student to add print statements or do a hand-trace focusing on a certain section of the code. Herein we present a software tool which, given a C++ program, some sample inputs, and respective expected outputs, uses discriminative graph mining to identify which lines in the program are most likely the source of a bug. Additionally, the particular operators (relational, logical, and arithmetic) that are used in the code may be considered in recommending where the bug may be. The tool includes a visual display of the control flow graph for each test case, allowing the user to step through the statements executed.**

Keywords-debugging; graph; data mining; visualization

I. INTRODUCTION

As discussed in [1], instructors and teaching assistants for introductory programming courses frequently are asked by their students: why doesn't my program work? Often the students understand the problem they are trying to solve well enough to articulate a variety of input and output scenarios. For example, if they are trying to find the sum of all even values in a list of numbers, they know that the input list {1, 2, 3, 4, 5} should produce a result of 6, and the input list {1, 3, 5, 7} should produce a result of 0. However, they frequently lack the ability to identify, or even narrow down, where a bug is occurring in their code when it does not produce the correct results. The recommendation to add print statements, although easy for experienced programmers, can require some skill and practice to master, and the use of a full-feature debugger can be cumbersome and intimidating to a novice programmer.

Herein we present *BugHint*, a software tool which, given a C++ program, some sample inputs, and respective outputs, uses discriminative graph mining to identify which lines in the C++ program are most likely causing the erroneous results. Additionally, the particular relational, logical, and arithmetic operators that are used in the program may be considered since beginning programmers tend to make more semantic errors with certain operators and in expressions that utilize multiple operators. The tool includes a visual display of the control flow graph for each test case (i.e., sample input), allowing the user to step through the statements as they are executed. The goal is that the student will take the bug hint and subsequently scrutinize the logic and code in the identified section of the program, thereby finishing the debugging process on his/her own.

The organization of this paper is as follows. Section II provides a brief overview of related work in debugging experiences with beginning programmers and the use of visualization in debugging. Section III discusses the foundation for and implementation of our software tool in terms of the graph mining analysis algorithms. Section IV presents the graphic user interface. A summary and conclusions are given in Section V. Future work is discussed in Section VI.

II. RELATED WORK

A. Debugging Experiences with Beginning Programmers

Several studies (e.g., [1], [2], and [3]) have identified problems that students experience with coding in introductory computer science courses, resulting in a proliferation of program bugs. Debugging strategies such as strategically placed print statements can be difficult to teach [1]. There are full-feature debugging tools such as GDB, which allow one to set breakpoints in the code and/or watch the values of variables change during execution of the program. However, for some novice programmers these tools can be too cumbersome and/or intimidating to use. After years of study, there is no consensus as to whether beginning programmers should be exposed to a full-feature debugger.

There have been studies that have successfully integrated the teaching of programming and a debugger at the introductory level. In [2] the authors used a debugger to demonstrate construction of Java objects and function calls in addition to using the debugger to find bugs in programs. Similarly, the authors of [4] used debugging exercises and simple debugger

DOI Reference Number: 10.18293/VLSS2018-029

functions to reinforce programming concepts (e.g., loops) that they were teaching.

However, full-feature debugger tools are not without criticism. In addition to the complaint that they may further confound the debugging experience for novice programmers who are already dealing with learning about an editor, operating system commands, compiler error messages, and programming language syntax, there is the issue that debuggers can potentially introduce additional bugs. A heisenbug is a software bug that is introduced when one attempts to study or analyze a program. Running a program in a debugger can actually modify the original code, changing memory addresses of variables and the timing of the execution. Debuggers often provide watches or other user interfaces that cause additional code to be executed, which, in turn, modify the state of the program. Time also can be a factor in heisenbugs, because race conditions may not occur when the program is slowed down by single-stepping through lines of code with the debugger.

Many visual debugging tools (such as DDD [5], Nemiver [6], or those debugging tools built into IDEs) provide a more user-friendly interface to command line debugging tools. Command line debugging tools suffer from the limitations of the interface; viewing where execution has stopped or paused requires programmer intervention, determining where breakpoints are to be placed (or have been placed) can be difficult (often requiring the programmer to figure out the exact line number at which they want to break), and determining the path that the execution followed can be difficult if breakpoints are not set appropriately. Additionally, if the programmer wants to figure out how their code behaves with multiple inputs, they will need to change the code, recompile, and run the debugging tool again. For veteran programmers this task is routine (and often more tedious than difficult); for novice programmers the complexity and power of these tools can be daunting and difficult to grasp.

Herein we do not seek to answer the question of whether the use of a full-feature debugger should be integrated into an introductory programming course. Rather, it is our intention to present a simple tool which the student can use as a debugging aid and training tool. Our aim is similar to the function of the instructor or teaching assistant who provides a hint as to where in the student's code the bug might be occurring. It is still up to the student to add print statements, do a hand-trace focusing on those particular statements, or use other techniques to try to fix the problem on his/her own, considering various input-output test cases.

B. *Visualization in Debugging*

Many contemporary debugging tools provide some type of visual representation of the source code in addition to displaying the program as text. This visual representation could be in the form of a flow chart (e.g., Visustin [7]), a control flow graph (e.g. KDevelop [8] and Dr. Garbage [9]), or UML diagrams (e.g., Eclipse ObjectAid [10]). The objective of the visualization is to facilitate understanding of some properties of the program such as the logic and/or the interactions between code blocks. To this end, animation (not just a static representation) of program execution has long been found to be useful.

Just as UML diagrams were deemed to be particularly helpful for object-oriented programming languages like Java and C++, control flow graphs have been found to be useful in debuggers for various programming paradigms. The authors of [11] presented GRASP, a graphical environment for analyzing Prolog (i.e., logic) programs; the tool dynamically animates the executed sequence of Prolog subgoals as a control flow graph and allows the user to inspect instantiation of variables as s/he steps through the execution. In [12] the authors introduced a debugging tool for MPI (i.e., parallel) programs that displays a message-passing graph of the execution of an MPI application; parts of the graph are hidden or highlighted based on the sequence of MPI calls that occur during a particular execution. Mochi [13] was created as a visual debugging tool for Hadoop (i.e., distributed programs); it displays the control flow of the workloads of each processor as a graph, allowing the user to observe the map and shuffle processing that takes place, and possibly identify erroneous sequencing and/or data partitioning.

III. IMPLEMENTATION

A. *Discriminative Graph Mining*

Our tool, *BugHint*, was motivated by the work presented in [14] for identifying bug signatures using discriminative graph mining. The basic idea is to first produce a control flow graph for a program written in a procedural programming language (in our case, this is C++). In brief, a control flow graph is a directed graph made up of nodes representing basic blocks. Each basic block contains one or more statements from the program. There is an edge from basic block Bi to basic block Bj if program execution can flow from Bi to Bj. For more information on control flow graphs and determination of basic blocks, see [15]. For C and C++ programs, a control flow diagram can be generated by compiling the program with *clang* and *opt* (we specify no optimization), and then creating the graph as a dot graph description language file using *dot*.

As an example, consider the C++ program shown in Fig. 1 which is supposed to replace only the first occurrence of either *x* or *y* in an array *a* with the value of *z*. This program does not perform that task correctly; it contains a bug. For simplicity, the code to output the final values of the array is commented out in this program since it is not where the bug occurs.

An example of a control flow graph for this program is shown in Fig. 2. In this graph there are eight blocks; the figure shows which lines of code are contained in each block.

After constructing a control flow graph for the program to be analyzed, our tool needs to consider test cases. These need to be specified in terms of sample input and expected output. The test cases should be as representative as possible of all boundary conditions for the program. However, a novice programmer may be unfamiliar with that notion. At the very least, the user must specify at least one input sample that is known to produce correct output and at least one input sample that is known to produce incorrect output; the user must distinguish these as 'correct' and 'incorrect.' In Table 1 we list some sample test cases for the example program shown in Fig. 1.

```
int main( )                         // line 1
{
  // inputs to the program
  int x = 1;
  int y = 7;
  int z = 0;
  int a[2] = {1, 2};
  int arraySize = 2;

  for (int i = 0; i < arraySize; i++)  // line 2
  {
    if (a[i] == x)                  // line 3
    {
      a[i] = z;                     // line 4
    }                               // line 5
    if (a[i] == y)                  // line 6
    {
      a[i] = z;                     // line 7
    }                               // line 8
  }                                 // line 9
  // code to output a[ ] ...
  return(0);                        // line 10
}
```

Figure 1. Example C++ program

For each sample case, our tool produces a code trace in terms of the lines executed for the specified input. The code traces for the four sample cases shown in Table 1 are listed in Table 2. It should be noted that if there is an infinite loop (which is a common bug) during execution of one of the sample input cases, the output from the code trace should be sufficient to identify the line(s) where the bug is occurring and no further analysis should be necessary. From each code trace, we also generate a control flow graph for that sequence. The control flow graphs for code traces 1 and 2 from Table 2 are shown in Fig. 3; the control flow graphs for code traces 3 and 4 are the same as the graph shown in Fig. 2.

The collection of graphs for the sample cases is next analyzed to identify non-discriminative edges. A non-discriminative edge is an edge that appears in every graph that is in the collection of execution graphs. Such edges are removed from each graph in the collection since they are the same in each execution, and, as such, are not informative in distinguishing where the bug occurs. The collection of control flow graphs with non-discriminative edges removed for our running example is shown in Fig. 4. Finally, the collection of graphs is analyzed to determine what subgraph is common to the faulty (i.e., incorrect output) execution graphs, but not common to the correct execution graphs. This corresponds to the section of code where the bug likely occurs. For our running example, such a discriminative control flow graph is shown in Fig. 5. It tells us that the bug involves blocks B4, B6, and B7, which correspond to lines 4-8 in the program. The hope is that the student will use this information to realize that, after changing the value to *z* in line 4, the program should not proceed to lines 6-8 since the specifications of the problem were to change either, not both, the occurrence of *x* or *y* to *z*.

The discriminative graph, and hence the bug in the program, may not consist of lines that are executed in the incorrect cases, but not executed in the correct cases (as was the situation in this example program); it could be the reverse situation. Or it could be the case that we cannot find a subgraph that is common to *all* faulty (or correct) execution graphs, but not common to the correct (or faulty) execution graphs. The algorithms we utilize for identifying the "best" discriminative graph are explained next. These differ slightly from those proposed in [14] and [16] for discriminative graph mining.

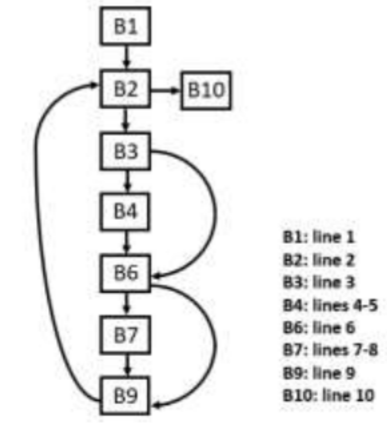


Figure 2. Control flow graph for the example program

TABLE 1. SAMPLE TEST CASES FOR THE EXAMPLE PROGRAM (INCORRECT CASES HIGHLIGHTED

Sample No.	a	x	y	z	Result
1	{1, 2}	1	7	0	{0, 2}
2	{1, 2}	7	1	0	{0, 2}
3	{1, 7}	1	7	0	{0, 0}
4	{1, 7}	7	1	0	{0, 0}

TABLE 2. CODE TRACES FOR THE EXAMPLE PROGRAM

	Trace Line Numbers	Trace Block Numbers
1	1 2 3 4 5 6 9 2 3 6 9 2 10	B1 B2 B3 B4 B6 B9 B2 B3 B6 B9 B2 B10
2	1 2 3 6 7 8 9 2 3 6 9 2 10	B1 B2 B3 B6 B7 B9 B2 B3 B6 B9 B2 B10
3	1 2 3 4 5 6 9 2 3 6 7 8 9 2 10	B1 B2 B3 B4 B6 B9 B2 B3 B6 B7 B9 B2 B10
4	1 2 3 6 7 8 9 2 3 4 5 6 9 2 10	B1 B2 B3 B6 B7 B9 B2 B3 B4 B6 B9 B2 B10

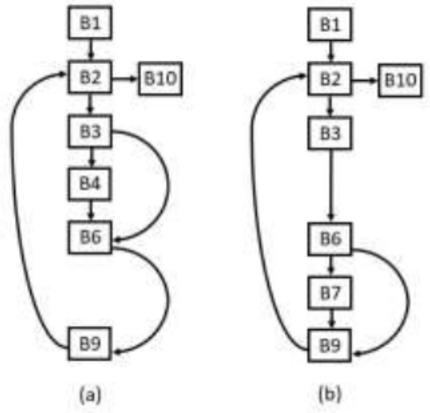


Figure 3. Control flow graphs for (a) trace 1 and (b) trace 2 from Table 2

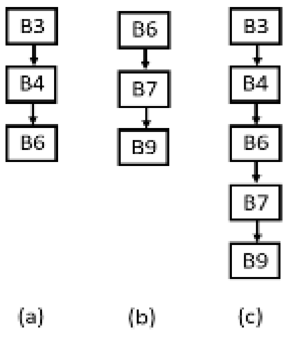


Figure 4. Control flow graphs with non-discriminative edges removed for (a) trace 1, (b) trace 2, and (c) traces 3 and 4 from Table 2

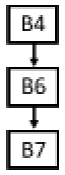


Figure 5. Discriminative control flow graph for the example program

Let C^+ and C^- represent the sets of control flow graphs for the sample test cases producing correct and incorrect results, respectively; we require that there be at least one graph in each such set. The function *FindDiscriminativeGraph* (Fig. 6) first removes non-discriminative edges from the graphs in both sets. It then calls *CreateDiscriminativeGraph* (Fig. 7) to try to find a subgraph that is common to all faulty execution graphs, but not common to all the correct execution graphs. If we are unable to find such a graph, then the function *RelaxedCreate-DiscriminativeGraph* (Fig. 8) is called, which relaxes the requirement that the subgraph we seek not be present in all of the correct execution graphs; instead the subgraph only has to not be present in $\alpha * |C^+|$ of the correct execution graphs, where α is a user-specified parameter (our default is $\alpha = 0.5$).

FindDiscriminativeGraph and *CreateDiscriminativeGraph* use a function called *Augment*; this function takes the subgraph G and adds to it an edge (and possibly a node) such that the source vertex exists in G, and the edge (and destination node) exists in all graphs in S1. In this way, a subgraph with an additional edge that exists in all elements of S1 is created and considered by the algorithm.

If we still fail to find a discriminative subgraph, then the bug likely does not involve code that is executed in all faulty cases and not in correct cases, but rather involves code that is executed in correct cases and not in faulty cases. Thus, we again call *CreateDiscriminativeGraph*, but reverse the order of the parameters (C^+ and C^-) from our previous call. If we still fail to find a discriminative subgraph, we again call *Relaxed-CreateDiscriminativeGraph* and look for a subgraph that only has to not be present in $\beta * |C^+|$ of the correct execution graphs, where β is a user-specified parameter (our default is $\beta = 0.5$).

It is possible that the resulting discriminative graph will be disconnected. We output the smallest connected component in that graph using the assumption that a novice programmer will want to focus on a single, sequential section of his/her program for scrutinizing the bug, rather that examining multiple, “fragmented” sections of code.

It should be noted that it is possible that our algorithm will not find any graph that meets the discriminative conditions. This could be because the specified test cases do not adequately exercise all paths through the control flow graph or it could be that the path through the control flow graph will be the same regardless of the input. Additionally, it could be the case that multiple subgraphs could be viable candidates to be the source of the bug. These last two situations are addressed in the next section.

```
Algorithm: FindDiscriminativeGraph(C+, C-, α, β)
C+: set of control flow graphs for inputs producing correct output
C-: set of control flow graphs for inputs producing incorrect output
α: percentage of graphs that discriminative subgraph need not be present in C+ when relaxing conditions
β: percentage of graphs that discriminative subgraph need not be present in C- when relaxing conditions
1.  remove non-discriminative edges from graphs in C+ and C-;
2.  G = CreateDiscriminativeGraph(C-, C+);
3.  if G is empty then
4.    G = RelaxedCreateDiscriminativeGraph(C-, C+,
                                            |C+| * α);
5.    if G is empty then
6.      G = CreateDiscriminativeGraph(C+, C-);
7.      if G is empty then
8.        G = RelaxedCreateDiscriminativeGraph(C+, C-,
                                                |C-| * β);
9.      end-if;
10.   end-if;
11. end-if;
12. G' =  smallest connected component in G;
13. output G'
```

Figure 6. Algorithm for *FindDiscriminativeGraph*

```
Algorithm: CreateDiscriminativeGraph(S1, S2)
S1: set of control flow graphs
S2: set of control flow graphs
1.  FreqSG = queue of 1-edge subgraphs in S1;
2.  while FreqSG is not empty do
3.    G = FreqSG.dequeue();
4.    if G is not in any graph in S2 then
5.      return(G);
6.    end-if;
7.    NewGraphs = Augment(G);
8.    for each graph G’ in NewGraphs do
9.      FreqSG.enqueue(G’);
10.   end-for;
11. end-while;
12. return(empty graph)
```

Figure 7. Algorithm for *CreateDiscriminativeGraph*

```
Algorithm: RelaxedCreateDiscriminativeGraph(S1, S2, γ)
S1: set of control flow graphs
S2: set of control flow graphs
γ: threshold for number of graphs discriminative subgraph
must be present in
1.  FreqSG = queue of 1-edge subgraphs in S1;
2.  while FregSG is not empty do
3.     G = FreqSG.dequeue();
4.     if G is in < γ graphs in S2 then
5.        return(G);
6.     end-if;
7.     NewGraphs = Augment(G);
8.     for each graph G' in NewGraphs do
9.        FreqSG.enqueue(G');
10.    end-for;
11. end-while;
12. return(empty graph)
```

Figure 8. Algorithm for *RelaxedCreateDiscriminativeGraph*

B. *Operator Complexity*

Earlier versions of *BugHint* [21] focused solely on the discriminative graphs. This approach is effective when the erroneous code causes the execution to follow a different code path in the CFG. However, it is often the case with novice programmers that, in an attempt to be clever, they introduce some relatively complex calculation in the condition of a loop or an if-statement that effectively short-circuits the branch and forces the code to always execute the same branches. In our testing of *BugHint*, we identified cases where no basic blocks were identified as problematic despite the presence of a bug. As such, the decision was made to augment the CFG analysis with more information based on additional scrutinization.

Semantic errors frequently occur in lines of code that are more complex in structure. Unless the errors encountered are in output formatting, a simple C printf statement or C++ cout statement is unlikely to introduce a semantic error into a program. Instead, bugs are often introduced in lines of code that compare or change the values of variables in the program. The more complex a line (or basic block) is in terms of relational, logical, and arithmetic operators, the more likely it is that a sematic error will occur. Additionally, some operators like && or ||, while familiar to seasoned programmers, seem foreign in their functionality to new coders. *BugHint* takes this into account by looking not only at the CFG analysis of the program (with good and bad traces) but at the complexity of each basic block encountered in the program.

Since there is no published formal study analyzing the frequency of semantic errors for various relational, logical, and arithmetic operators by beginning programmers, the complexity of various C++ operators was determined by over 40 years of collective teaching and tutoring experience of the authors for *BugHint* consideration. Several frequently used operators were sorted into ranks which were used to create a complexity score for each operator. Table 3 shows the ranks of these operators as well as the (informal) rationale for their relative ranking. A lower ranking indicates that the operator is less likely to cause a semantic error when used in an expression (on its own). For example, +, *, and – correspond to familiar arithmetic concepts for beginning programmers; students have used those operations since grade school. It should be noted that division (/) is not grouped with addition, multiplication, and, subtraction because integer division (e.g., 3 / 4 evaluates to 0 in C++) is the source of many semantic errors for novice programmers. The comparison operator == also is a frequent source of bugs since many students confuse it with the assignment operator =. In contrast to multiplication, subtraction, and addition, the logical operators && and || are new concepts to most beginning programmers and require higher order thinking than addition, multiplication, and subtraction; thus, they are ranked much higher. If not explicitly listed in the table, an operator defaults to a rank of 0.

The complexity score of an operator is defined as its rank /5.0, or more generally:

$$complexity = rank\ /\ number\_of\_ranks$$

This results in a complexity score in the range [0, 1) that increases as the complexity of an operator increases. The generalization of the calculation of the complexity is such that the ranks and relative complexities can be fine-tuned based on empirical testing and observation.

TABLE 3. OPERATOR COMPLEXITY RANKS

Rank	Ops	Rationale
0	+ * -	Most familiar
1	< > <= >= ! += -= *=	Easy concepts, but more foreign than rank 0
2	== /	== confused with =, / is integer division
3	++ -- %	New concepts, pre/post increment/decrement can give different results
4	&& ||	Requires higher order thinking

The complexity of a basic block is calculated as the sum of the number of operators in the basic block (a non-negative integer) and the average complexity of an operator in the basic block (a real value less than 1). This ensure that basic blocks with more operators are indicated as more complex, but, between basic blocks of the same number of operations, those with higher average operator complexity are classified as more complex; these are indications of two common Code Smells [19]. If the basic block has relatively few lines of code but a high number of operations, this may indicate an excessively long line of code (or a God Line); if the basic block has a large number of lines, it may hint at the presence of the Long Method code smell. While these code smells are not an indication that a bug is present, they are frequent indicators that something is wrong, or that a refactoring may result in code that is easier to debug and maintain.

BugHint augments the CFG analysis of source code with the complexity analysis. After identifying basic blocks that could be problematic using the discriminative graph mining, the basic blocks that were identified as potentially problematic are sorted by complexity. The top N (a user-specified parameter that defaults to 2) are reported to the user as the most likely to contain the bug.

This augmentation to the CFG analysis has some benefits. Consider the following main function (where here the value of *answer* is hard-coded on the second line instead of being entered by the user to allow for *BugHint* to operate on the source):

```
// block 1
bool validEntry;
char answer = 'n';
cout << "Is this yes or no?" << endl;
validEntry = (answer == 'y' && answer == 'Y') ||
             (answer == 'n' && answer == 'N');
// block 2
if (validEntry)
   cout << "Valid entry!" << endl;
// block 3
else
   cout << "Invalid Entry" << endl;
// block 4
return 0;
```

Veteran programmers will quickly notice that the condition in line 4 is incorrect and will always return false regardless of the value of *answer*. However, CFG analysis will not indicate erroneous blocks because all traces will contain blocks 1, 3, and 4 in that order. Examination of the complexity of the basic blocks will show that blocks 2 through 4 have a complexity of 0, but block 1 has a complexity of 3 + (0.8 * 3) / 3 = 3.8. *BugHint* would augment the CFG analysis (which would be unable to find a discriminative subgraph) with the information from the complexity analysis and indicate to the user that the problem exists in block one.

IV. User Interface

Some of the tools used to generate the information needed to display to the user are not easily installable/usable on all platforms (specifically, clang/LLVM and some bash tools are not easily installable on Windows). Making the *BugHunt* platform independent was a top priority so that it could be readily available to as broad a range of novice programmers as possible. *BugHunt* has been developed as a web application written in Node.js and using vis.js for the visualization of the control flow graphs. The web application backend interfaces with various utilities written in Python, which make the appropriate system calls to clean, format, instrument, and run the source code with various starting conditions. Additionally, the backend analyzes the traces and basic blocks of the provided source code.

Web applications inherently have a large number of security concerns, and a system designed specifically to compile and run arbitrary C/C++ code exhibits an exceptionally large attack surface area. This application was not designed to run as a globally hosted web application, but rather was intended to be deployed to individuals using emerging technologies such as Docker (for platform-independent deployments of the web application that run locally) or the Linux Subsystem for Windows (which would allow a more native desktop interface to be quickly developed using Electron Shell [20]).

Initially, the user is presented with a single toolbar with a button that allows them to upload a single C++ source file. When the system receives the file, it generates the CFG using clang's analysis tools (specifically DumpCFG), calculates the complexity of each basic block, maps lines of code to basic blocks, and instruments the code by adding output statements at the beginning of each basic block (see Fig. 9). The system then runs the instrumented program and sends the resulting traces back to the UI. A student can choose individual starting configurations and walk forwards and backwards through the execution (with lines of code highlighted) using arrow buttons on the UI.

```
#include <iostream>
using namespace std;
int main()
{
    int a = 0;
    double d = 1.1;
    for (int i=0; i<10; i++)
    {
        cout << "hi" << endl;
    }
    return 0;
}
```

Figure 9a) Original C++ source

```
#include <iostream>
#include <sstream>
#include <unistd.h>
void __bbinstr(const char bbid[]) {
 std::cerr << bbid << std::endl;
 return;
}
#include <iostream>

using namespace std;
/*:B6:*/ int main() /*:/B6:*/
{
 /*:B5:*/ __bbinstr("B5");
 int a = /*%a%*/ 0 /*%~a%*/; /*:/B5:*/
 /*:B5:*/ double d = /*%d%*/ 1.1 /*%~d%*/;
/*:/B5:*/
 for (int i = 0; __bbinstr("B4"), i < 10;
__bbinstr("B2"), i++) {
   /*:B3:*/ __bbinstr("B3");
   cout << "hi" << endl; /*:/B3:*/
 }
 /*:B1:*/ __bbinstr("B1");
 return 0; /*:/B1:*/
}
```

Figure 9b) Instrumented C++ source (formatted for more readability)

The system assumes that the initial configuration that was uploaded by the student produced a good trace. However, the user can add additional starting conditions (by changing the starting values of the variables at initialization) and specify whether or not it produces a good trace (i.e., correct results) or a bad trace (i.e., incorrect results). At any point, the student may select "Get Hint." The UI will then send the traces to the backend, which will analyze the CFG, the traces, and the complexity of the basic blocks. The resulting list of basic blocks is sent to the GUI for display.

The CFG analysis requires at least one “good” trace. While the student might not understand why their code is producing the correct answer in some cases, it must be assumed that the novice programmer has managed to get correct output in at least one case. As use of this tool requires knowledge of inputs and expected outputs (and thus what inputs result in correct output), it can be reasonably assumed that the student can identify at least one scenario where their code produces a correct result.

Fig. 10 shows a screenshot of the main view in the *BugHunt* GUI with the example program from Fig. 1 and the test cases from Table 1. The hints have been augmented using the complexity information. The arrow buttons in the GUI allow the user to step forwards and backwards through a selected execution case; both the corresponding nodes and (text) lines in the program subsequently will be highlighted. Any particular block in an execution sequence (listed below the graph display) also can be selected (i.e., clicked-on) with the mouse.

BugHint also supports user-defined functions. When loading a single cpp source file containing multiple functions, the system splits the information so that the CFG for each function and the code for each function displays in a tab. When walking through the traces, the tabbed view switches automatically to the function which is currently being executed in the trace. Fig. 11, 12, and 13 show this tabbed view for an implementation of a function that finds the maximum of two integer values; the function contains a bug (Fig. 13). When giving hints, *BugHunt* will highlight all lines of code that are suspected of being in the vicinity of the bug (i.e., are in the detected block).

V. Summary and Conclusions

Herein we have presented a simple debugging tool, which, given a C++ program that has a logic error just serious enough to occasionally produce erroneous output while sometimes producing correct output, and some sample inputs with corresponding outputs, uses discriminative graph mining to identify which lines in the program are most likely the source of the bug. Additionally, it may examine the particular relational, logical, and arithmetic operators that are used in the code to determine what lines in the code are probably causing the bug. The tool includes a visual display of the control flow graph for each test case, allowing the user to step through the statements executed.

In a previous study [21], we found that students who completed pre-training using *BugHint* did better on post-testing than students who completed pre-training without *BugHint*, even though all groups had no help for post-training. During a post-training exercise where both groups completed the exact same activity of debugging three practice programs, the treatment group found more of the bugs, self-generated more informative test cases and reasoning regarding those test cases, and self-generated more helpful comments to add to the code itself. Hence, it appears that the extra-formalized method of using *BugHint* may improve the way students think about the debugging practice.

We look forward to utilizing *BugHint* in our introductory programming courses, and performing additional usefulness and usability studies to guide further refinement of this tool, and reduce some of the time that novice students spend debugging their programs.

VI. FUTURE WORK

As with any visual programming environment, we expect that there will be a challenge in accommodating the additional visual complexity in the graphical user interface that will result from larger programs. We intend to perform usefulness and usability studies with novice programmers to find ways of implementing visual representation and navigation of a fairly large number of modules of the control flow graph in a manner that they (the students) can best understand. By fairly large number we mean a number commensurate with the size of programs that beginning programmers write. From our own teaching experience that has been approximately 50 lines of code (with no user-defined functions) at the beginning of the CS1 semester, and 1500 or more lines of code (with 20 or more user-defined functions) by the end of the semester.

We also intend to compare our system to the hints that would be produced by utilizing other existing algorithms for finding discriminative subgraphs (e.g., [17] and [18]) and/or adding other options (in addition to our current α and β parameters) to our algorithm in order to find the best discriminative subgraph, and hence provide the best suggestion for the bug hint.

Additionally, the system currently has rather strict rules on the formatting of the files it can handle due to the nature of the parser. Years of introductory computer science education experience has demonstrated that these rules are frequently not followed by students despite the insistence of course instructors. The backend will be made more robust to reduce these restrictions.

References

[1] C. Lewis and C. Gregg, “How Do You Teach Debugging?: Resources and Strategies for Better Student Debugging”, Proceedings of the 47th ACM Technical Symposium on Computing Science Education, Memphis, TN, Mar. 2-5, 2016, p. 706.

[2] R.C. Bryce, A. Cooley, A. Hansen, and N. Hayrapetyan, “A One Year Empirical Study of Student Programming Bugs”, Frontiers in Education Conference, Washington, DC, Oct. 27-30, 2010, pp. 1-7.

[3] J.H.I.I. Cross, T.D. Hendrix, and L.A. Barowski, “Using the Debugger as an Integral Part of Teaching CS1”, “Frontiers in Education, Boston, MA, Nov. 6-9, 2002. pp. 1-6.

[4] G.C. Lee and J.C. Wu, “Debug It: A Debugging Practicing System”, Computers & Education, 32, 1999, pp. 165-179.

[5] DataDisplayDebugger, https://www.gnu.org/software/ddd/

[6] Nemiver, https://wiki.gnome.org/Apps/Nemiver

[7] Visustin, http://www.aivosto.com/visustin.html

[8] KDevelop, https://liveblue.wordpress.com/2009/07/21/3-visualize-your-code-in-kdevelop/

[9] Dr. Garbage, https://sourceforge.net/projects/drgarbagetools/files/

[10] Eclipse ObjectAid, http://www.objectaid.com/sequence-diagram

[11] H. Shinomi, "Graphical Representation and Execution Animation for Prolog Programs", International Workshop on Industrial Applications of Machine Intelligence and Vision (MIV-89), Tokyo, Apr. 10-12, 1989, pp. 181-186.

[12] B. Schaeli, A. Al-Shabibi, and R.D. Hersch, "Visual Debugging of MPI Applications", in Recent Advances in Parallel Virtual Machine and Message Passing Interface, A. Lastovetsky, T. Kechadi, J. Dongarra (eds)., EuroPVM/MPI, Lecture Notes in Computer Science, vol. 5205, Springer, Berlin, Heidelberg, 2008, pp. 239-247.

[13] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Mochi: Visual Log-Analysis Based Tools for Debugging Hadoop", CMU-PDL-09-103, Parallel Data Laboratory, Carnegie Mellon University, Pittsburg, PA, May 2009.

[14] H. Cheng, D. Lo, Y. Zhou, X. Wang, and X. Yan, "Identifying Bug Signatures Using Discriminative Graph Mining", ISSTA, Chicago, IL, Jul. 19-23, 2009, pp. 141-151.

[15] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman, Compilers: Principles, Techniques, and Tools, Addison Wesley, 2nd edition, 2006.

[16] X. Yan, H. Cheng, J. Han, and P.S. Yu, "Mining Significant Graph Patterns by Leap Search", SIGMOD 2008, Jun. 9-12, 2008, Vancouver, BC, Canada, pp. 433-444.

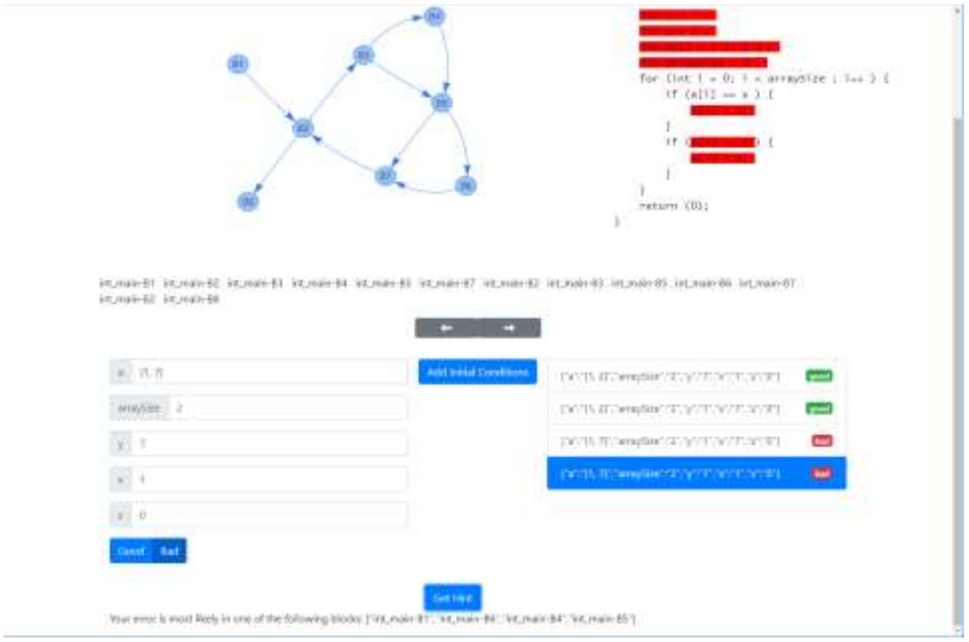
[17] N. Jin and W. Wei, "LTS: Discriminative Subgraph Mining by Learning from Search History", IEEE 27th International Conference on Data Engineering (ICDE), 2011, pp. 207-218.

[18] M.G.A. El-Wahab, A.E. Aboutabl, and W.M.H. El Behaidy, "Graph Mining for Software Fault Localization: An Edge Ranking Based Approach", Journal of Communications Software and Systems, Vol. 13. No. 4, Dec. 2017, pp. 178-188.

[19] Fowler, M., Beck, K., Brant, J., Opdyke, W. and Roberts, D., 1999. *Refactoring improving the design of existing code*. Addison-Wesley Professional. pp. 76-78.

[20] Electron, https://electronjs.org/

[21] J.L. Leopold, N.W. Eloe, and P. Taylor, "BugHint: A Visual Debugger Based on Graph Mining", Proceedings of the 24th International Conference on Visualization and Visual Languages, San Francisco, CA, June 29-30, 2018, pp. 109-118

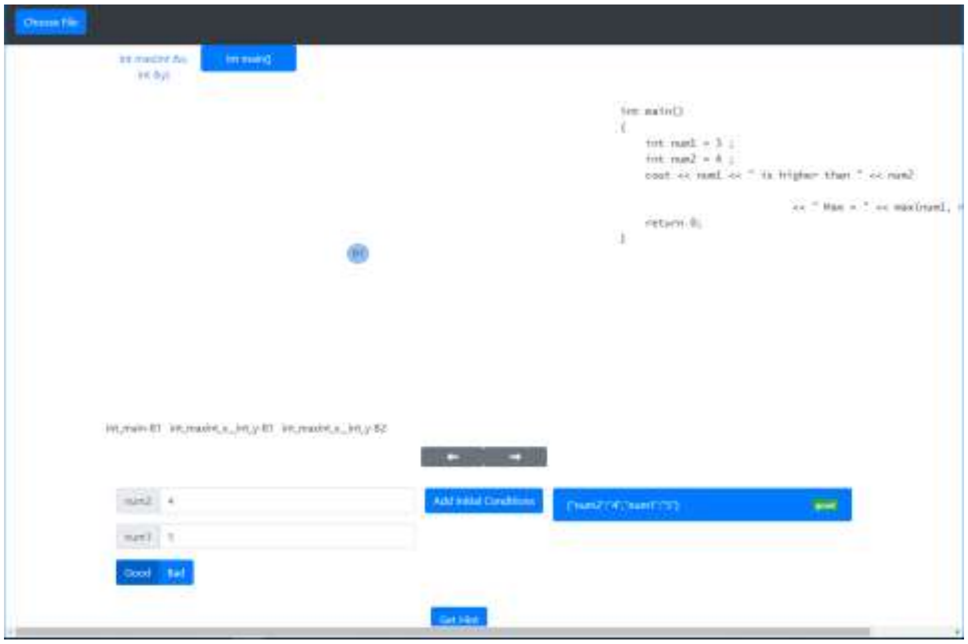


```
[illegible]
[illegible]
[illegible]
[illegible]
for (int i = 0; i < arraySize ; i++ ) {
    if (a[i] == x ) {
        [illegible]
    }
    if ([illegible]) {
        [illegible]
    }
}
return (0);
}
```

int_main-B1 int_main-B2 int_main-B3 int_main-B4 int_main-B5 int_main-B7 int_main-B2 int_main-B3 int_main-B5 int_main-B6 int_main-B7 int_main-B2 int_main-B8

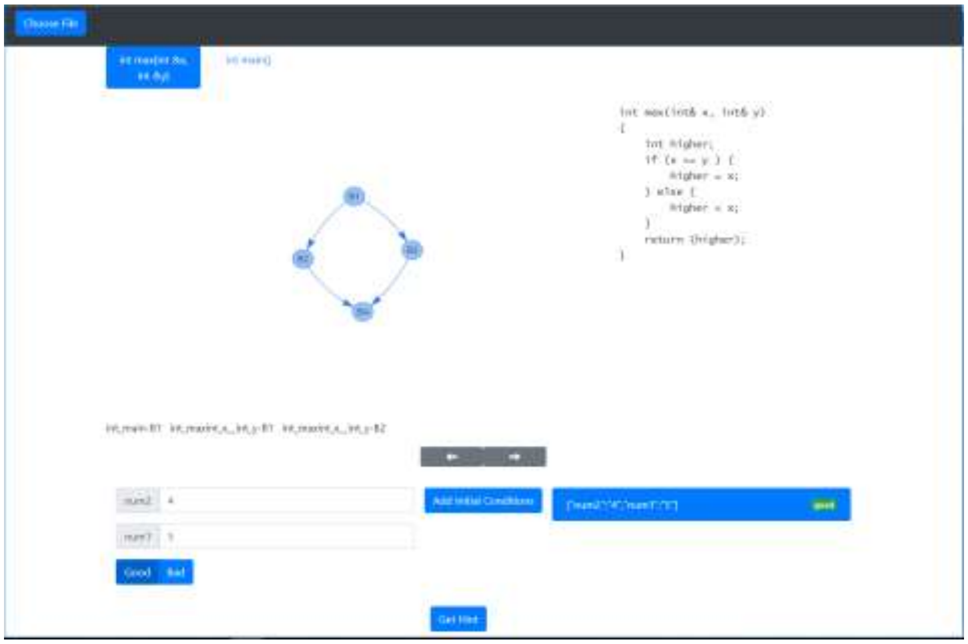
Your error is most likely in one of the following blocks: ['int_main-B1', 'int_main-B6', 'int_main-B4', 'int_main-B5']

Figure 10. *BugHunt* GUI, showing the possible erroneous lines as determined by control flow graph trace analysis and complexity analysis



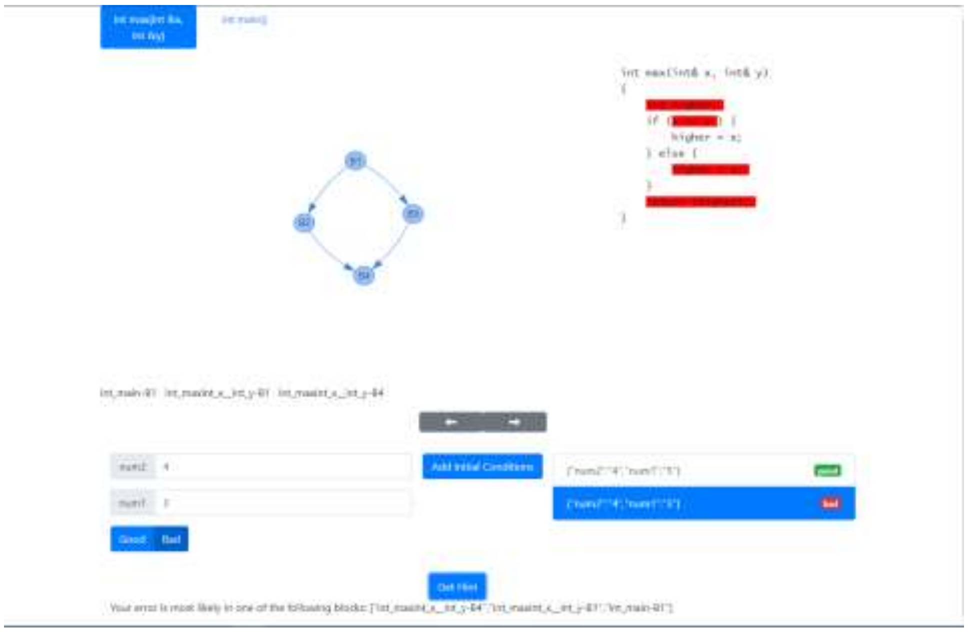
```
int main()
{
    int num1 = 3 ;
    int num2 = 4 ;
    cout << num1 << " is higher than " << num2

                                   << " Max = " << max(num1,
    return 0;
}
```

Figure 11. *BugHunt* GUI after loading a file that calls a user-defined function



```
int max(int& x, int& y)
{
    int higher;
    if (x >= y ) {
        higher = x;
    } else {
        higher = x;
    }
    return (higher);
}
```

Figure 12. *BugHunt* GUI when switching function views



```
int max(int& x, int& y)
{
    [illegible]
    if ([illegible]) {
        higher = x;
    } else {
        [illegible]
    }
    [illegible]
}
```

{"num2":"4","num1":"3"}

{"num2":"4","num1":"3"}

Your error is most likely in one of the following blocks: ["int_max(int_x,_int_y-B4","int_max(int_x,_int_y-B1","int_main-B1"]

Figure 13. Hint to the location of a bug in max function

Edge-based graph grammar: theory and support system

Xiaoqin Zeng[1] Yufeng Liu[1] Zhan Shi[1] Yingfeng Wang[2] Yang Zou[1] Jun Kong[3] Kang Zhang[4]

[1]Institute of Intelligence Science and Technology, Hohai University, Nanjing, Jiangsu, China

[2]School of Information Technology, Middle Georgia State University, Macon, GA 31206, USA

[3]Department of Computer Science, North Dakota State University, Fargo, ND 58102, USA

[4]Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA

Abstract—**As a useful formal tool, graph grammar provides a rigorous but intuitive way for defining graphical languages and analyzing graphs. This paper presents a new context-sensitive graph grammar formalism called Edge-based Graph Grammar or EGG, in which a new methodology is proposed to tackle issues, such as the embedding problem, the membership problem and the parsing algorithm. It presents the formal definitions of EGG and its language with a proof of its decidability. Then, a new parsing algorithm with an analyses of its computational complexity is given for checking the structural correctness or validity of a given host graph. The paper finally describes the development of an EGG support system with friendly GUI.**

Keywords-component; graph grammar; graphical language; embedding problem; parsing; production rule

I. INTRODUCTION

With the development of human-computer interaction techniques, graphical languages have been applied to various application domains, such as modeling visual interaction processes [1, 2], designing graphical user interface in multimedia applications [3], visual queries to databases [4], and defining the layout of a GUI in multimedia applications [3]. Conceptually, objects described by graphical languages can be abstracted as graphs consisting of nodes and edges. For the specification and analysis of these types of graphs, graph grammars [5, 6] are an ideal formal and intuitive tool.

It is well-known that formal string grammar lays a solid theoretical foundation for the definition and parsing of programming languages. For the same reason, graphical languages also need the corresponding formal graph grammars. Compared with string grammar, graph grammars set a theoretical basis to visual languages [7]. However, the implementation of a graphical language is usually not as easy as implementing string languages [8]. This is mostly due to the fact that the extension from one-dimensional string grammars to two-dimensional graph grammars raises new issues [9] such as the embedding problem, the membership problem, high parsing complexity.

There have been a number of graph grammars and their applications in the literature [10-27]. According to the type of grammatical productions, graph grammars could be mainly divided into two categories: context-free and context-sensitive. The main differences between the two are the production formation and the expressive power. On the one hand, a context-free grammar requires that only a single non-terminal node be allowed on the left-hand side of a production [16]. In early years, many context-free grammars were proposed [17-21]. Since the productions of these graph grammars are quite simple, their expressive power is limited, which hinders the scope of their applications. On the other hand, in response to the increasing demands of intricate graph-oriented applications, researchers have developed several context-sensitive graph grammars, such as PLC (picture layout grammar) [21], CMG (constrain multiset grammar) [22], LGG (layered graph grammar) [23], RGG (reserved graph grammar) [8], SGG (spatial graph grammar) [24, 25]. These context-sensitive graph grammars allow the left-hand side of a production to be a graph rather than a node, so bring more expressive power. LGG and RGG are the most representatives of context-sensitive graph grammars.

Rekers and Schürr [23] proposed a context-sensitive graph grammar formalism called *Layered Graph Grammar* (LGG) for defining and parsing graphical visual languages. First, productions in LGG differ widely from others by introducing context nodes that are not replaced in a derivation or reduction operation. Second, to solve the embedding problem, LGG puts a restriction on the definition of a redex in a host graph by requiring its nodes that are isomorphic to non-context nodes in productions can only link to other nodes in the host graph that are isomorphic to the context nodes in the productions. This restriction ensures no creation of dangling edges when a redex in a host graph is replaced. Third, a very intricate layer decomposition constraint is introduced to solve the membership problem.

Based and improved on LGG, Zhang et al. [8] proposed another context-sensitive grammar called *Reserved Graph*

DOI Reference Number: 10.18293/VLSS-028

Grammar (RGG), which defines the structure of graphs by introducing a two-level structure for each node as a super-vertex containing sub-vertices connected with edges. In addition, RGG introduces a marking mechanism to tackle the embedding problem, in which a unique label is used to identify all context elements. Further, with the introduction of selection-free productions to graph grammars, a Selection-Free Parsing Algorithm (SFPA) is designed for a selection-free RGG, which only needs to consider one parsing path and thus can efficiently parse graphs with polynomial time complexity [8]. Later on, Kong et al. [24, 25] extended RGG by introducing spatial notations and mechanisms. The spatial specifications of the extended RGG, called *Spatial Graph Grammar* (SGG), can qualitatively express the spatial relationships among objects and reduce the parsing complexity using the spatial information.

Both LGG and RGG have been applied widely to the definition, analysis and transformation of visual languages [28-37], such as Visual XML Schemas [29, 30], Design Pattern Evolution and Verification [32, 33], Generic Visual Language Generation Environments [28]. However, they still have deficiencies. For example, the LGG's context nodes and layer decomposition constraint make productions difficult to design. RGG's two-level node structure and marking mechanism are not intuitive and make them difficult to apply to general graphs.

This paper presents our work on the improvements over the existing graph grammars with the following contributions.

- A new context-sensitive graph grammar formalism called EGG, which uses edges instead of nodes to concisely express the context in productions for simply and efficiently solving the embedding problem.
- A size-increasing constraint applied to the structure of productions for solving the membership problem, easing the design of productions.
- A new general parsing algorithm for checking the structural correctness and validity of given host graphs; and the implementation of an EGG graph grammar support system, which provides friendly GUI for end users to design and apply graph grammars.

The rest of the paper is organized as follows. Section 2 presents graphical and grammatical preliminaries, introducing new terms used in Section 3, which gives the formal definitions of EGG and its language with a proof of its decidability. Section 4 presents a parsing algorithm and its complexity analysis. Section 5 describes the developed EGG support system. Finally, Section 6 concludes the paper.

II. Graphical and Grammatical Preliminaries

In node-edge graphs, a node typically represents an abstract object and an edge represents some kind of relationship between two connected nodes. Each node n in a node set N can be connected with none or more edges, and each edge e in an edge set E is only connected with two nodes. An edge can be directed or undirected depending on whether it has a direction between the two connected nodes. Because an undirected edge can be treated as two directed edges with reverse directions, without loss of generality this paper only considers directed edges.

In string grammars, labels play an important role as identifiers, and so do labels in graph grammars. Let L be a finite set of labels. Depending on the usage of a label, L can further be divided into terminal label set L_T, nonterminal label set L_{NT}, and mark label set L_M, namely $L \quad L_T \cup L_{NT} \cup L_M$, $L_T \cap L_{NT} \quad \Phi$, and $L_M \cap (L_T \cup L_{NT}) \quad \Phi$.

By combining the techniques of both graph theory and formal language, we introduce a series of new definitions and notations here.

Definition 2.1 $\mathrm{n} \quad (\mathrm{l})$ is a node with label l in a given finite label set L.

Definition 2.2 $\mathrm{e} \quad (\mathrm{n_s}, \mathrm{n_e})$ is a directed edge, where

- $\mathrm{n_s}$ is the start node of the directed edge;
- $\mathrm{n_e}$ is the end node of the directed edge.

Based on the above definitions of node and directed edge, we further introduce the following notations:

- $\mathrm{E_s}$ is a set of directed edges starting from a node;
- $\mathrm{E_e}$ is a set of directed edges ending to a node;
- $\mathrm{d(n)}$ is the degree indicating the number of directed edges connected to n, i.e. $\mathrm{d(n)} \quad |\mathrm{E_s} \cup \mathrm{E_e}|$;
- $\mathrm{d_s(n)}$ is the out-degree indicating the number of directed edges starting from n, i.e. $\mathrm{d_s(n)} \quad |\mathrm{E_s}|$;
- $\mathrm{d_e(n)}$ is the in-degree indicating the number of directed edges ending to n, i.e. $\mathrm{d_e(n)} \quad |\mathrm{E_e}|$.

Obviously, $d(n) \quad d_s(n) + d_e(n)$. For simplicity, notations like $n.l$ and $n.E_s$ express the corresponding components of node n, and are applicable to other definitions throughout this paper.

Unlike an undirected edge, a directed edge needs to distinguish start node and end node. Besides, an edge may also carry a label for clear identification.

Definition 2.3 $\mathrm{G} \quad (\mathrm{N}, \mathrm{E})$ is a graph on given label set L, where

- N is a node set that is associated with a two-way partition into $\mathrm{N_T}$ and $\mathrm{N_{NT}}$, the elements of $\mathrm{N_T}$ are called terminal nodes and the elements of $\mathrm{N_{NT}}$ are called non-terminal nodes;
- E is a directed edge set with $\mathrm{E} \subseteq \mathrm{N} \times \mathrm{N}$.

We then have the following mappings for mathematically expressing grammatical items.

- $\mathrm{f_{NL}}: \mathrm{N} \rightarrow \mathrm{L}$, a mapping from node n to label $\mathrm{l} \in \mathrm{L}$, i.e., $\mathrm{f_{NL}(n)} \quad \mathrm{n.l}$;
- $\mathrm{f_{EN_s}}: \mathrm{E} \rightarrow \mathrm{N}$, a mapping from directed edge e to its start node, i.e., $\mathrm{f_{EN_s}(e)} \quad \mathrm{e.n_s}$;
- $\mathrm{f_{EN_e}}: \mathrm{E} \rightarrow \mathrm{N}$, a mapping from directed edge e to its end node, i.e., $\mathrm{f_{EN_e}(e)} \quad \mathrm{e.n_e}$.

In EGG, dangling edge set $\dot{E}$ is introduced to represent contexts, in which each edge is connected with only one node being either a start or end node, namely $\dot{E} \quad \dot{E}_s \cup \dot{E}_e$ with $\dot{E}_s$ $\{\dot{e}_s | \dot{e}_s \quad (n_s, \emptyset)\}$, $\dot{E}_e$ $\{\dot{e}_e | \dot{e}_e \quad (\emptyset, n_e)\}$ and $\dot{E}_s \cap \dot{E}_e \quad \Phi$. In addition to dangling edges, a marking mechanism is also introduced to mark dangling edges. The concepts of dangling edge and marking mechanism solves the embedding problem in EGG. Fig. 1 illustrates a graph including dangling edges with

$\dot{E}$ {1,2,3}. The graph is called a *dangling edge graph* and can be defined as follows.

Definition 2.4 G $(\mathrm{N}, \mathrm{E}, \mathrm{M})$ is a *dangling edge graph* on given label set L, in which,

- N is a node set;
- E is an edge set including dangling edges, which is associated with a two-way partition into E and $\dot{\mathrm{E}}$;
- $\mathrm{M} \subseteq \mathrm{L_M}$ is a mark set for marking dangling edges to distinguish different contexts.

Essentially, G is an extension of G by introducing dangling edge and G can be regarded as a special case of G. Similarly, there is an extra mapping as follows.

- $\mathrm{f_{EM}}: \dot{\mathrm{E}} \rightarrow \mathrm{M}$, an injective mapping from dangling edge $\dot{\mathrm{e}}$ to its mark m, i.e., $\mathrm{f_{EM}}(\dot{\mathrm{e}})$ m.

Note that dangling edge set $\dot{\mathrm{E}}$ may be empty, which leads to the empty corresponding mark set *M* and mapping f_{EM}. Based on the above defined dangling edge graph, a grammatical production can be defined as follows.

Definition 2.5 A production p is the expression $\mathrm{G_L} \coloneqq \mathrm{G_R}$, which consists of a left dangling edge graph $\mathrm{G_L}$ and right dangling edge graph $\mathrm{G_R}$ satisfying $\mathrm{G_L.M}$ $\mathrm{G_R.M}$.

In a production, dangling edges represent contexts and each pair of corresponding dangling edges between the left and right graphs are labeled by a unique mark to maintain their corresponding relationship. Using dangling edges and their corresponding marks, the replacement of a redex by either a left or right graph in a production can be done without ambiguity. In some special cases, a wildcard dangling edge is needed to represent an arbitrary number of edges, e.g., one entity may be connected with any number of attributes in an entity relationship diagram. For simplicity and without generality, the concept of wildcard edge is not discussed here. Fig. 2 is an example of a set of EGG productions specifying a process flow diagram with $\{\mathrm{begin, assign, fork, join, send, receive, if, endif}\} \subseteq \mathrm{L_T}$ and $\{\mathrm{stat}\} \subseteq \mathrm{L_{NT}}$.

The function of a production is to transform a graph to another graph. However, the transformation needs to satisfy some conditions in which isomorphism is fundamental.

Definition 2.6 Graphs G and Q are isomorphic, denoted as $\mathrm{G} \approx \mathrm{Q}$, $\mathrm{f_{NL}}$ and $\mathrm{f_{NL}}'$ are two mappings for G and Q respectively, if and only if there exist two bijective mappings $\mathrm{f_{NN}}: \mathrm{G.N} \leftrightarrow \mathrm{Q.N}$ and $\mathrm{f_{EE}}: \mathrm{G.E} \leftrightarrow \mathrm{Q.E}$, and the following are satisfied:

- $\forall \mathrm{n}(((\mathrm{n} \in \mathrm{G.N}) \vee (\mathrm{n} \in \mathrm{Q.N})) \rightarrow (\mathrm{f_{NL}}(\mathrm{n})$ $\mathrm{f_{NL}}'(\mathrm{f_{NN}}(\mathrm{n}))))$;
- $\forall \mathrm{e}(((\mathrm{e} \in \mathrm{G.E}) \vee (\mathrm{e} \in \mathrm{Q.E})) \rightarrow (\mathrm{f_{NN}}(\mathrm{f_{EN_s}}(\mathrm{e}))$ $\mathrm{f_{EN_s}}(\mathrm{f_{EE}}(\mathrm{e}))) \wedge (\mathrm{f_{NN}}(\mathrm{f_{EN_e}}(\mathrm{e}))$ $\mathrm{f_{EN_e}}(\mathrm{f_{EE}}(\mathrm{e}))))$.

An isomorphism between two graphs means that their corresponding nodes have the same label, and the same out-degree and in-degree. In addition, the corresponding edges have the same start and end nodes.

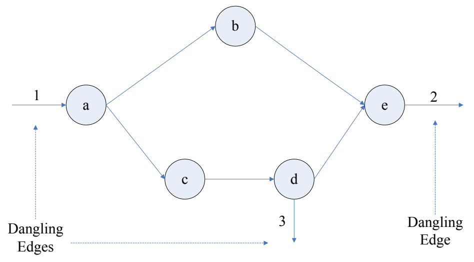


Figure 1. A dangling edge graph

Definition 2.7 Graph Q is the sub-graph of G, denoted as $\mathrm{Q} \in \mathrm{Sub(G)}$, if and only if the following are satisfied:

- $(\mathrm{Q.N} \subseteq \mathrm{G.N}) \wedge (\mathrm{Q.E} \subseteq \mathrm{G.E})$.

Graph Q is a sub-graph of G means that Q is part of G.

Definition 2.8 Graph Q is the core graph of G, denoted as Q Cor(G), if and only if the following is satisfied:

- $(\mathrm{Q.N}$ $\mathrm{G.N}) \wedge (\mathrm{Q.E}$ $(\mathrm{G.E}$ $\mathrm{G.\dot{E}})) \wedge (\mathrm{Q.L}$ $\mathrm{G.L})$.

Core graph Q is the sub-graph of graph G obtained by removing all dangling edges from graph G and keeping all the nodes and non-dangling edges of graph G. The graph in Fig. 3 is the core graph of that in Fig. 1.

Definition 2.9 If graph Q is a sub-graph of graph G and may include dangling edges, and $\mathrm{G_{L\ R}}$ is a graph being left or right side of a production, Q is a redex of G with respect to $\mathrm{G_{L\ R}}$, denoted as $\mathrm{Q} \in \mathrm{Redex}(\mathrm{G}, \mathrm{G_{L\ R}})$, if and only if there exits bijective mappings $\mathrm{f_{NN}}: \mathrm{Q.N} \leftrightarrow \mathrm{G_{L\ R}.N}$ and $\mathrm{f_{EE}}: \mathrm{Q.E} \leftrightarrow \mathrm{G_{L\ R}.E}$, and the following are satisfied:

- $\mathrm{Cor(Q)} \approx \mathrm{Cor(G_{L\ R})}$;
- $\forall \mathrm{n}((\mathrm{n} \in \mathrm{Q}) \rightarrow (\mathrm{d_s}(\mathrm{n})$ $\mathrm{d_s}(\mathrm{f_{NN}}(\mathrm{n}))) \wedge (\mathrm{d_e}(\mathrm{n})$ $\mathrm{d_e}(\mathrm{f_{NN}}(\mathrm{n}))))$.

To explain the above definition, we provide an example in the following three figures. Fig. 4 is graph $\mathrm{G_{L\ R}}$, and Fig. 5 is a given host graph G. Obviously, graph Q in Fig. 6 is the sub-graph of G. According to *Definition 2.9*, Q is a redex of G with respect to $\mathrm{G_{L\ R}}$.

In host graph G, if there is sub-graph Q being the redex of G with respect to $\mathrm{G_{L\ R}}$ that is a left or right side graph of a production, then one could use the right or left side graph of the production to replace Q in G. This process is called *graph transformation* or *replacement*, as formally defined below.

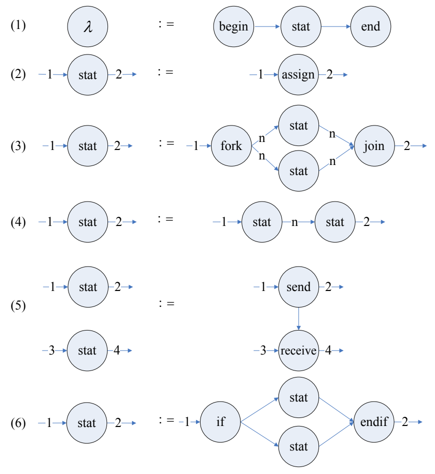


Figure 2. A set of EGG productions

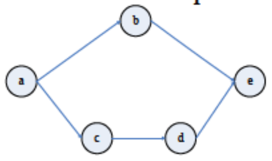


Figure 3. The core graph of the graph in Figure 1

Definition 2.10 An *L-application* to graph G is a transformation that generates graph G' using production p: $G_L := G_R$, denoted as G' $\mathrm{Tr}(G, Q, G_L, G_R)$, where $Q \in \mathrm{Redex}(G, G_L)$, and $\mathrm{Cor}(G_R)$ is used to replace Q in G. The L-application is also called *derivation* operation and denoted as $G \rightarrow^{p} G'$.

If a sequence of L-applications for graph G is: $G \rightarrow^{p} G_1', G_1' \rightarrow^{p_2} G_2', \ldots, G_{n\ 1}' \rightarrow^{p_n} G_n'$, then $G \rightarrow^{*} G_n'$ can be used to concisely express this process.

Definition 2.11 An *R-application* to graph G is a transformation that generates graph G'' using production p: $G_L := G_R$, denoted as G'' $\mathrm{Tr}(G, Q, G_R, G_L)$, where $Q \in \mathrm{Redex}(G, G_R)$, and $\mathrm{Cor}(G_L)$ is used to replace Q in G. The R-application is also called *reduction* operation and denoted as $G \mapsto^{p} G''$.

Similar to L-applications, a sequence of R-applications, which is $G \mapsto^{p} G_1'', G_1'' \mapsto^{p_2} G_2'', \ldots, G_{n\ 1}'' \mapsto^{p_n} G_n''$, can be expressed as $G \mapsto^{*} G_n''$.

Fig. 7 shows a derivation process from an initial graph using the productions in Fig. 2.

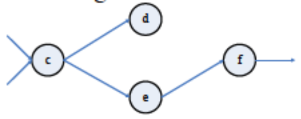


Figure 4. A graph $G_{L\ R}$ with dangling edges

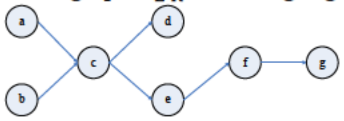


Figure 5. A host graph G

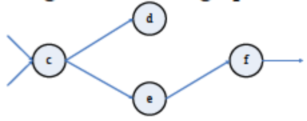


Figure 6. The sub-graph Q is a redex of G with respect to $G_{L\ R}$

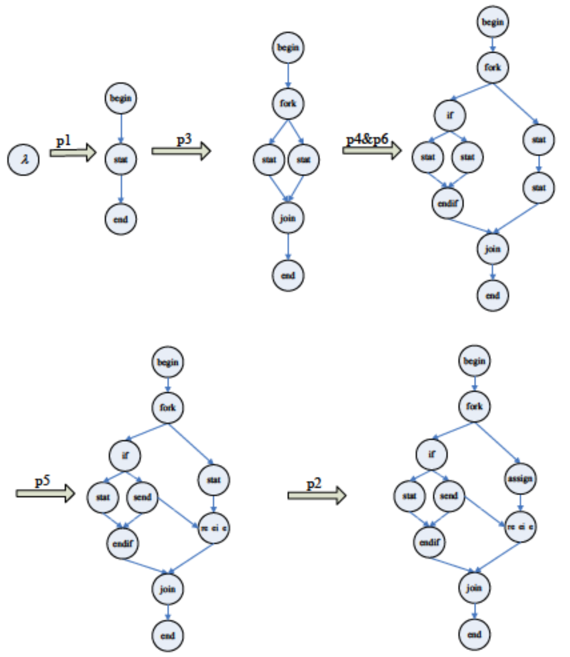


Figure 7. A graph L-application process using EGG productions

The formal definition of EGG and its language are discussed below.

III. AN Edge-based Graph Grammar Formalism

To solve the embedding and membership problems, EGG employs edges rather than nodes in the two sides of a production to directly express contexts and introduces a size-increasing constraint to ensure the decidability of EGG.

3.1 Definition of EGG and its language

Based on the definitions in Section 2.1, an edge-based context-sensitive graph grammar formalism and its language can be defined as follows.

Definition 3.1 An EGG is a 3-tuple（λ, L, P), where:

- λ is an initial graph;
- L is a label set containing terminal and non-terminal labels, i.e., L $L_T \cup L_{NT}$;
- P is a set of productions, and each production $p \in P$ in the form of $G_L := G_R$ must satisfy the following constraints:
  (1) λ must be a left graph of a production;
  (2) G_R must be nonempty;
  (3) The size of left graph must be no more than that of right graph, i.e., $|G_L.N| \leq |G_R.N|$. If they are equal, the number of terminal nodes in left graph must be less than that of right graph, i.e., $|G_L.N_T| < |G_R.N_T|$.

Similar to string grammars, graph grammars with arbitrary graphs on the left and right sides of productions may face the membership problem, that is, their languages are not decidable in general. EGG introduces a size-increasing constraint for each production to solve the membership problem. The constraint ensures that any given host graphs can be parsed with EGG productions within a finite number of R-applications. Also, the constraint is weak with little impact on the flexibility of context-sensitive grammars and easier to implement than that of LGG and RGG for grammar designers.

Theoretically, a graph grammar is a formal tool for rigorously defining a graph language, which is a set of graphs that can be derived from the initial graph. Below is the formal definition of a graph language.

Definition 3.2 Let egg (λ, L, P) be a grammar of EGG, its language $\Gamma(\text{egg})$ can be formally defined as $\Gamma(\text{egg})$ $\{G|(\lambda \to^{*} G)\wedge(f_{NL}(G.N) \subseteq L_T)\}$.

Practically, a graph grammar is a useful tool for automatically analyzing graphs' validity. If a given graph can be reduced to the initial graph with a finite series of R-applications of a graph grammar, this graph is regarded as belonging to the grammar's language. Otherwise, the graph does not belong to the graph language or the graph grammar is not decidable.

3.2 Decidability of EGG

When an EGG is given, its language is determined. It is decidable whether an arbitrarily given graph is in the language or not because of the support of the following theorem.

Theorem 1. For EGG egg (λ, L, P) and arbitrary nonempty graph G, it is decidable whether or not G is in $\Gamma(\text{egg})$.

Proof: For arbitrarily given graph G with a finite number of terminal nodes, a sequence of graphs can be generated in an R-application process starting from G. Because of the size-increasing constraint and the number of nodes in the graph G being finite, the R-application process cannot execute circularly and must stop in finite steps, namely, $G \mapsto^{*} G_n$ and G_n being unable to reduce any more by R-application. Further, the number of such sequences without a loop is also finite. Thus it is feasible to enumerate all such sequences and check whether $G \mapsto^{*} G_n$ and G_n λ are held for at least one of the sequences. If there exists one, then $G \in \Gamma(\text{egg})$, otherwise $G \notin \Gamma(\text{egg})$.

In the proof, the size-increasing constraint on the productions of EGG guarantees the decidability of EGG because the constraint requires that each R-application should at least either remove a node or change a terminal node to a non-terminal node in the reduced graph. Therefore, R-application can only be applied finite times to any host graph of a given size.

IV. Parsing algorithm of EGG

Generally, a graph grammar needs to be equipped with a parsing mechanism for automatically checking whether a given graph, called *host graph*, is structurally correct or valid with respect to the graph language defined by the grammar. Having proved that the membership problem is decidable for EGG in the previous section, this section presents a parsing algorithm, which checks if a host graph can be reduced to the initial graph by applying the EGG grammar's productions to perform a series of R-applications. A parsing algorithm usually needs to incorporate the following three interrelated actions:

- Search in the host graph for the redexes of a production's right graph;
- Perform an R-application with a found redex to generate a new host graph from the current host graph; and
- Trace all the R-application paths by applying in turn the above two actions until a path leading to the initial graph is found or all possible paths have been exhausted.

In the following, the above three actions are discussed in more detail. The first is the searching. The second is the R-application. Finally, the tracing combines the two to perform a parsing.

4.1 Search for redexes

A procedure for searching all redexes is given below, which takes host graph G and right graph G_R as input and returns a set of redexes found.

```
FindRedexForRight(Graph G, Graph $G_R$)
{
  RedexSet = $\Phi$;
  G-Nodes = OrderNodeSequence(G);
  $G_R$-Nodes = OrderNodeSequence($G_R$);
  CandidateNodeSet = FindCandidateSet(G-Nodes, $G_R$-Nodes);
  for each Candidate $\in$ CandidateNodeSet
    RedexSet = RedexSet $\cup$ GenerateRedex(Candidate, G, $G_R$);
  return(RedexSet);
}
```

In the procedure, function OrderNodeSequence sequences the nodes in host graph G and right graph G_R separately according to their labels' alphabetical order. Function FindCandidateSet finds all possible node sequences from G-Nodes as candidates under the condition that all nodes in a candidate have the corresponding nodes in the G_R-Nodes of the same degree. Function GenerateRedex generates all possible redexes derived from a candidate. Note that a candidate with all nodes plus their connected edges including dangling edges, notated as C_G, only has the same structure as G_R, and may generate more than one redex. This is because a node in G_R may have more than one dangling edge in the same direction and different matches of the dangling edges between C_G and G_R may generate different redexes. Fig. 8 illustrates a case that uses the marking mechanism, where Fig. 8(a) is host graph G containing C_G in a dotted box, and Fig. 8(b) is a production containing the corresponding right graph G_R. Since the node labelled 'b' in G_R is connected with two outgoing dangling edges, there are two ways of assigning the marks numbered '1' and '2' to $C_G.\dot{E}$, and thus two redexes are generated accordingly as illustrated in Fig. 8(c) and Fig. 8(d). Fig. 8(e) and Fig. 8(f) demonstrate that the two redexes are different and can reduce graph G to two different graphs.

4.2 R-application

A procedure for performing an R-application is given below, which takes host graph G, redex Q, and production p relevant to Q as inputs and generates a reduced graph.

```
RightApplication(Graph G, redex Q, Production p)
{
    AddMark(p.$G_R$, G);
    InsertLeftGraph(p.$G_L$, G);
```

```
        DeleteRedex(G, Q);
        G′=ClearMark(G);
        return(G′)
}
```

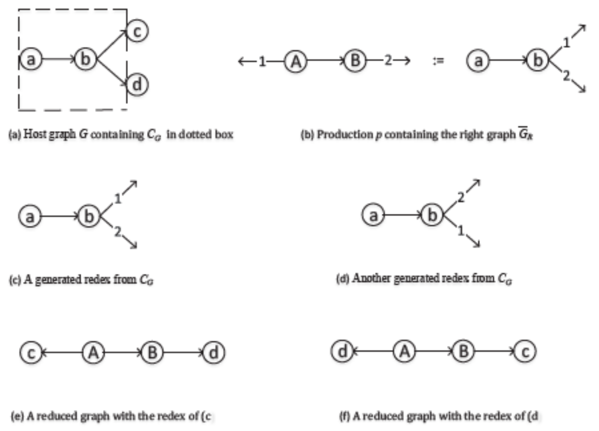

(a) Host graph G containing C_G in dotted box
(b) Production p containing the right graph $\overline{G}_R$
(c) A generated redex from C_G
(d) Another generated redex from C_G
(e) A reduced graph with the redex of (c
(f) A reduced graph with the redex of (d

Figure 8. Reductions with two different redexes generated from a C_G

In R-application, function AddMark adds all the dangling edges' marks of $p.G_R$ to G according to the edge mapping between them. Function InsertLeftGraph inserts $p.G_L$ into G by connecting all dangling edges of $p.G_L$ to the corresponding nodes in G according to the marks added previously. Function DeleteRedex deletes redex Q from G . Finally, function ClearMark clears all added marks in G to generate reduced graph G′.

4.3 Parsing

Based on the above discussions, it is now feasible to trace all possible R-application paths starting from a given host graph to check if there exists one path that leads to the initial graph. The tracing needs to maintain a mapping between a redex and its host graph for performing the corresponding R-application. As such a mapping is usually many to one, the tracing employs two stacks to separately store the redexes found and the intermediate host graph yielded, and employs a delimiter in the redex stack to delimit a group of redexes that correspond to the same host graph. The delimiter makes the correspondence manageable by synchronizing the contents in the two stacks. The function takes a graph and a set of productions as input and returns a definite answer indicating whether the graph is valid or not.

```
Parsing (Graph G, ProductionSet P)
{
 loop-1:  while (G ≠ λ)
          {
            DELIMITER → RedexStack;
            // push
 loop-2:    for all p ∈ P
            {
              RedexSet = FindRedexForRight(G, p.G_R);
 loop-3:          for all Redex ∈ RedexSet;
                      (Redex, p)→ RedexStack;
            // push
             }
            (Redex, p) ← RedexStack;
            // pop
 loop-4:    while (Redex = DELIMITER)
            {
               If (HostStack != NULL ∧ RedexStack != NULL)
              G← HostStack;
             // pop
             (Redex, p) ← RedexStack;
             // pop
              else
             return("Invalid");
             }
            HostStack ←G ;
            // push
            G = RightApplication(G, Redex, p);
          }
          return("Valid");
}
```

4.3.1 Time complexity

This subsection examines the time and space complexities of the above parsing algorithm.

Theorem 2. The time complexity of the parsing algorithm is $O(\left(\frac{n}{r}\right)^h (hh!)^r (d!\,h)^{rh}))$, where h is the number of nodes in the host graph to be parsed, r is the maximal number of nodes in the right graphs of all productions with d being the maximal number of dangling edges for each node, and n is the number of productions in the given EGG grammar.

Proof: According to the structure of the parsing algorithm, its maximal time complexity can be expressed as:

$$t \quad O(l_1(l_2(t_1 + l_3) + l_4 + t_2)),$$

where l_1 is the maximal number of iterations in the outermost loop-1, l_2 is the number of iterations in the first inner loop-2, l_3 is the number of iterations in the innermost loop-3, l_4 is the number of iterations in the second inner loop-4, and t_1 and t_2 are the worst time complexities of functions FindRedexForRight and RightApplication respectively.

In function FindRedexForRight, since the maximal possible number of selecting r nodes from h nodes is $A_h^r \quad h(h \quad 1) \ldots (h \quad r + 1)$, the worst complexity of searching for all candidates of a right graph in a given host graph is $O(h^r)$. Further, since a candidate may generate more than one redex due to different assignments of its dangling edges, the maximal possible number of actions to generate redexes from one candidate is $rA_d^d \quad rd!$. In fact, r and d can be considered constants when an EGG grammar is given. Thus, we have t_1 in $O(rd!\,h^r) \quad O(h^r)$.

In function RightApplication, since each of its sub-functions needs at most traversing the host graph once, thus it has t_2 $O(h)$.

As to the four iterations, l_2 needs first to be considered, which is in fact the number of productions, i.e., $l_2 \quad n$. l_3 is the number of redexes found in the host graph with respect to the right graph of a given production. Since the maximal number of candidates with respect to the right graph of a given production is $C_h^r \quad A_h^r/r!$ (by reasonably assuming $r \ll h$ for making sure that C_h^r is maximal) and each candidate can generate at most $(d!)^r$ redexes by taking dangling edges into consideration, it has $l_3 \leq (d!)^r C_h^r \quad O(h^r)$. Since $l_2 l_3$ is the total number of actions on pushing redexes into the redex stack and l_4 is the partial number of actions on popping redexes from the redex stack, l_4 should be no more than $l_2 l_3$, and thus can be ignored. The left is iteration number l_1, the worst case is when the algorithm's result is 'invalid', and all redexes found during parsing enter the stack. Each of the redexes, when popped out of the stack, leads to an iteration of the outmost loop. Therefore, l_1 is equal to the number of all redexes found.

An iteration of the outmost loop generates no more than $n(d!)^r C_h^r$ redexes for n productions and performs one R-application. According to the size-increasing condition, each R-application would reduce the size of the derived host graph. Since there are at most *h* R-applications that may not reduce the host graph size and an R-application would reduce the host graph size by at least 1, the following holds for l_1:

$$\begin{aligned}
L_1 \leq\ & (n(d!)^r C_h^r)^{h+1} n(d!)^r C_{h\ \ 1}^r n(d!)^r C_{h\ \ 2}^r \dots n(d!)^r \\
& \quad C_{h\ \ (h\ \ r\ \ 1)}^r n(d!)^r C_{h\ \ (h\ \ r)}^r \\
& (n(d!)^r)^{2h\ \ r+1} (C_h^r)^h C_{h\ \ 1}^r C_{h\ \ 2}^r \dots C_{r+1}^r C_r^r \\
& (n(d!)^r)^{2h\ \ r+1} (\frac{h}{(h\ \ r)\ r})^h \frac{(h\ \ 1)}{(h\ \ 1\ \ r)\ r} \dots \frac{(r+1)}{1\ r} \frac{r}{0\ r} \\
& (n(d!)^r)^{2h\ \ r+1} (\frac{h}{(h\ \ r)\ r})^h \prod_{u=1}^{h\ \ r\ \ 1} \frac{(u+r)}{r\ u} \\
& \frac{(n(d\ )^r)^{2h-r+}}{(r\ )^{2h-r-}} \left(h(h\quad 1)\dots(h\quad r+1)\right)^h \prod_{u=1}^{h\ \ r\ \ 1}(u+ \\
& \quad r)(u+r\quad 1)\dots(u+2)(u+1) \\
& O((n\frac{(d\ )^r}{r})^{2h} h^{rh} \prod_{u=1}^{h\ \ r\ \ 1} u^r) \\
& O((n\frac{(d\ )^r}{r})^{2h} h^{rh} ((h\quad r\quad 1)!)^r) \\
& O((n\frac{(d\ )^r}{r})^h (h^h h!)^r) \\
& O(\left(\frac{n}{r}\right)^h (h!)^r (d!\,h)^{rh}) \qquad (1)
\end{aligned}$$

Combining all the above discussions, one can finally obtain:

$$t \quad O(\left(\frac{n}{r}\right)^h (hh!)^r (d!\,h)^{rh}).$$

4.3.2 Space complexity

Theorem 3 The space complexity of the parsing algorithm is $O(h^{r+1})$, where *h* is the number of nodes in the host graph to be parsed, *r* is the maximal number of nodes in all the right graphs of productions.

Proof: The main space-consuming components are the redex stack and the host graph stack used in the parsing algorithm. We can, therefore, express the maximal space complexity as:

$$s \quad s_1 + s_2,$$

where s_1 is the space used by the redex stack and s_2 is the one by host graph stack. Without loss of generality, we can assume that the space taken by a redex is r and that by a host graph is h. Different from time complexity, the use of the stack space is not always increasing because pop operations would release space for reuse. Hence, the worst case is the maximal occupied space along with the longest R-application path, and the following holds for the redex stack and the host graph stack respectively.

$$\begin{aligned}
s_1 \leq\ & (rhnC_h^r(d!)^r + rnC_h^r(d!)^r + rnC_{h\ \ 1}^r(d!)^r + \cdots \\
& \quad + rnC_{h\ \ (h\ \ r)}^r(d!)^r) \\
& rn(d!)^r(hC_h^r + C_h^r + C_{h\ \ 1}^r + \cdots + C_r^r) \\
& rn(d!)^r(\frac{(h+1)}{(h\ \ r)\ r} + \frac{(h\ \ 1)}{(h\ \ 1\ \ r)\ r} + \cdots + \frac{r}{0\ r}) \\
& \frac{n}{(r\ \ 1)}(d!)^r(\frac{(h+1)}{(h\ \ r)} + \sum_{u=0}^{h\ \ r\ \ 1} \frac{(u+r)}{u}) \\
& \frac{n}{(r\ \ 1)}(d!)^r((h+1)h(h\quad 1)\dots(h\quad r+1) + \\
& \quad \sum_{u=0}^{h\ \ r\ \ 1}(u+r)(u+r\quad 1)\dots(u+1)) \\
& O(h^{r+1} + \sum_{u=0}^{h\ \ r\ \ 1} u^r) \\
& O(h^{r+1}); \\
s_2 \quad & hh + (h\quad 1) + \cdots + r \\
& O(h^2). \qquad (2)
\end{aligned}$$

Since $r \geq 1$, the following can be obtained:

$$s \quad O(h^{r+1}).$$

From the above analysis, we notice that the time complexity is extremely high while the space complexity is bounded by a polynomial factor. We also note that the structure of productions plays an important role in determining the algorithm's complexity. For example, if a stronger constraint such as $|p.G_L.N| < |p.G_R.N|$ is enforced on productions, then the first *h* R-applications that do not reduce the host graph size can be removed from (1). In addition, we find that the algorithm itself may be further improved to increase its efficiency, especially its average time cost. Moreover, like RGG, if the condition of Selection-Free [28] is satisfied, the Selection-Free Parsing Algorithm with polynomial time complexity can be used for EGG.

V. IMPLEMENTATION OF AN EGG SUPPORT SYSTEM

A graph grammar support system is a software platform that can be helpful for end users to easily use graph grammars. This section briefly describes the architecture and functions of an EGG support system, abbreviated as EGGSS.

From a user point of view, EGGSS supplies, besides normal GUI of Windows, extra graphical and grammatical tools to assist the user to draw graphs, design graph productions, define graph languages, perform graph transformations and parse graphs. They are visualized in a friendly fashion explained below.

- Graph Editor: performs all kinds of graph related operations, such as graph drawing, saving, deleting.
- Production Designer: for designing productions based on the Graph Editor, such as production generation and modification.
- Transformer: automatically performs L-application for transforming graph from one to another based on a given production.
- Language Definer: specifies labels, marks, etc. for defining the graph language via Productions and L-application.
- Parser: automatically performs a series of R-applications for checking the validity of a given graph.

Fig. 9 illustrates the end user view of EGGSS. Fig. 10 is an example window of EGGSS's user interface, where the upper row is the main menu with all operational items including not only graphical and grammatical operations but also other

Window GUI operations. On the left, a tree view allows users to manage XML files with saving, accessing and deleting operations. They can read graphs in XML format from the memory and save graph data to an XML file. On the right, the upper part shows an edited host graph and the lower part shows a designed production.

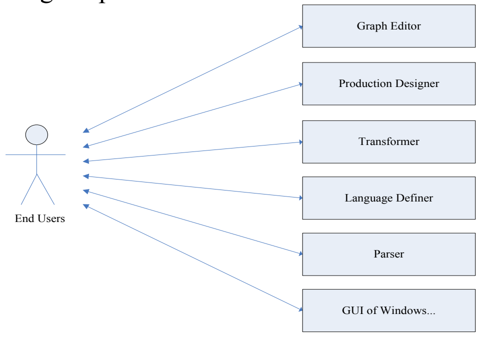


Figure 9. End user view of EGGSS

From a system point of view, EGGSS consists of basic modules, organized logically in layers to realize the system's grammatical functions explained below.

- Graph Transformation: automatically completes the transformation from one graph to another using productions. This module is essentially an L-application.
- Graph Parsing: performs grammatical analysis for a given host graph by automatically searching for all possible graph reduction operations to finally reduce to the initial symbol, namely the host graph is grammatically valid if and only if it could be reduced to the initial symbol. This module is essentially a series of R-applications.
- Graph Matching: finds redexes in a graph according to a given production.
- Graph Substitution: replaces a sub-graph in a given graph using the left or right graph of a production.
- XML Description: transfers graph expressions to XML descriptions and vice versa.

Fig. 11 shows the system architecture with relevant modules. In the architecture, three upper layers are implemented using C++ in the environment of Visual Studio 2005, while two lower layers are implemented using the existing XML open sources and software tools.

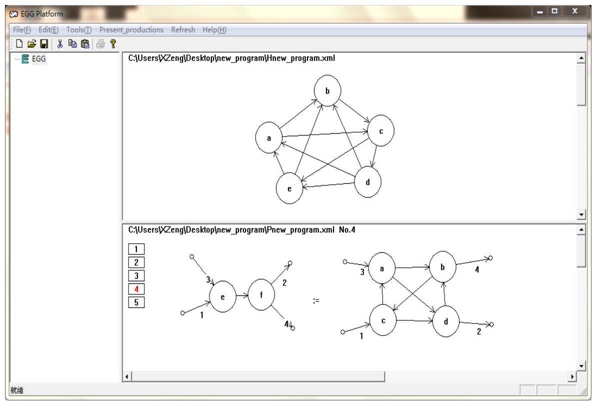

EGG Platform
File(F) Edit(E) Tools(T) Present_productions Refresh Help(H)
EGG
C:\Users\XZeng\Desktop\new_program\Hnew_program.xml
C:\Users\XZeng\Desktop\new_program\Pnew_program.xml No.4

Figure 10. A window of EGGSS's user interface

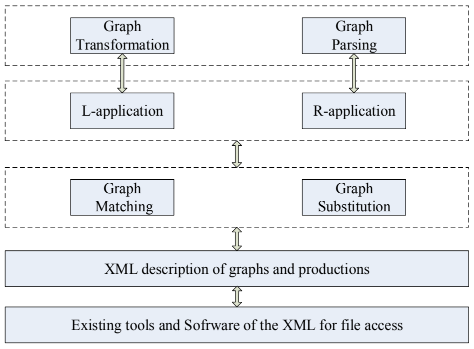

Graph Transformation
Graph Parsing
L-application
R-application
Graph Matching
Graph Substitution
XML description of graphs and productions
Existing tools and Sofrware of the XML for file access

Figure 11. The architecture of EGGSS

I. CONCLUSIONS

This paper has proposed a new graph grammar formalism, namely EGG, which aims at making the design and implementation of a graph grammar simple without weakening the expressive power of the grammar. The proposed EGG lays a solid foundation for a wide range of applications using graph grammars. Specifically, EGG focuses on tackling general graph languages and graph transformations with productions as simple as possible. First, EGG simplifies the expression of productions, in which the context nodes are eliminated and only edges linked to context nodes are kept. In this way, the structural information of graphs is still kept. Second, using dangling edges and their corresponding marks, the replacement of a redex by either a left or right graph in a production can be easily done without ambiguity. Third, the introduction of size-increase constraint to productions solves the membership problem, making EGG parsing algorithm terminable.

As a future research, we will attempt to find the way to reduce the parsing complexity, to further improve EGGSS to be friendlier for end users.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under grant 61170089.

REFERENCES

[1] P. Bottoni, S. K. Chang, M. F. Costabile, S. Levialdi, P. Mussio. On the specification of dynamic visual languages, Proc. IEEE Symposium on Visual Languages, 14-21, 1998.

[2] P. Bottoni, S. K. Chang, M. F. Costabile, S. Levialdi, P. Mussio. Modeling visual interactive systems through dynamic visual languages, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 32(6): 654-669, 2002.

[3] S. K. Chang. Extending visual languages for multimedia, IEEE Multimedia, 3(3): 18-26, 1996.

[4] S. K. Chang. A visual language compiler for information retrieval by visual reasoning, IEEE Transactions on Software Engineering, 16(10): 1136-1149, 1990.

[5] G. Rozenberg, H. Ehrig, Handbook of graph grammars and computing by graph transformation, Handb. Graph Grammars. 1 (1997) 1–8.

[6] H. Fahmy, D. Blostein, A Survey of Graph Grammars: Theory and Applications, in: IAPR Int. Conf. Pattern Recognit., 1992: pp. 294–298.

[7] C. Ermel, M. Rudolf, G. Taentzer, The AGG Approach: Language and Environment, in: Handb. Graph Grammars 2, 1999: pp. 551-603.

[8] D.-Q. Zhang, K. Zhang, J. Cao, A context-sensitive graph grammar formalism for the specification of visual languages, Comput. J. 44 (2001)

[9] X.-Q Zeng, K. Zhang, J. Kong, G.-L. Song, RGG+: An enhancement to the reserved graph grammar formalism, in: Proc. 2005 IEEE Symp. Vis. Lang. Human-Centric Comput., 2005: pp.272–274.

[10] D. Goik, K. Jopek, M. Paszyński, A. Lenharth, D. Nguyen, K. Pingali, Graph grammar based multi-thread multi-frontal direct solver with Galois scheduler, in: Procedia Comput. Sci., 2014: pp.960–969.

[11] L. Fürst, M. Mernik, V. Mahnič, Converting metamodels to graph grammars: doing without advanced graph grammar features, Softw. Syst. Model 14 (2013) 1297–1317.

[12] J. Heinen, C. Jansen, J.-P. Katoen, T. Noll, Verifying pointer programs using graph grammars, Sci. Comput. Program. 1 (2013) 7–12.

[13] Z. Shi, X.-Q. Zeng, S. Huang, H. Li, Transformation between BPMN and BPEL based on graph grammar, in: Proc. 5th Int. Conf. Comput. Commun. Netw. Technol., 2014: pp.1-6.

[14] F. Hermann, S. Gottmann, N. Nachtigall, H. Ehrig, B. Braatz, G. Morelli, A. Pierre, T. Engel, C. Ermel, Triple Graph Grammars in the Large for Translating Satellite Procedures, Theor. Prac. Model Transforms. 8568 (2014) 122-307.

[15] Y. Ong, K. Streit, M. Henke, W. Kurth, An approach to multiscale modelling with graph grammars, Ann. Bot. 114 (2014) 813–827.

[16] K. Wittenburg, L. Weitzman, Relational grammars: theory and practice in a visual language interface for process modeling. Vis. Lang. Theor. (1998) 193-217.

[17] G. Rozenberg, E. Welzl, Boundary NLC graph grammars-Basic definitions, normal forms, and complexity, Inf. Control. 69 (1986) 136–167.

[18] D. Janssens, G. Rozenberg, Graph grammars with neighbourhood-controlled embedding, Theor. Comput. Sci. 21 (1982) 55–74.

[19] F. Drewes, H.-J. Kreowski, A. Habel, Hyperedge Replacement Graph Grammars, in: Handb. Graph Grammars 1, 1997: pp.95–162.

[20] K. Wittenburg, Earley-style parsing for relational grammars, Proc. 8th IEEE Workshop. Vis. Lang., 1992: pp. 192-199.

[21] E. Golin, A Method for the specification and parsing of visual languages, PhD Thesis, 1991, Department of Computer Science, Brown University.

[22] K. Marriott, Constraint Multiset Grammars, Proc. IEEE Symp. Vis. Lang., 1994: pp. 118–125.

[23] J. Rekers, a Schürr, Defining and parsing visual languages with layered graph grammars, J. Vis. Lang. Comput. 8 (1997) 27–55.

[24] J. Kong, K. Zhang, X.-Q. Zeng, Spatial graph grammars for graphical user interfaces, ACM Trans. Comput. Interact. 13 (2006) 268–307.

[25] J. Kong, K. Zhang, Parsing Spatial Graph Grammars, Proc. 2004 IEEE Symp. Vis. Lang. Hum. Centric Comput. (2004) 99–101.

[26] M. Decker, H. Che, A. Oberweis, P. Stürzel, M. Vogel, Modeling mobile workflows with BPMN, in: ICMB GMR 2010 - 2010 9th Int. Conf. Mob. Business/2010 9th Glob. Mobil. Roundtable, 2010: pp.272–279.

[27] C. Kim, M. Ando, Node replacement graph grammars with dynamic node relabeling, Theor. Comput. Sci. 583 (2015) 40–50.

[28] K. Zhang, D.-Q. Zhang, J. Cao, Design, construction, and application of a generic visual language generation environment, IEEE Trans. Softw. Eng. 27 (2001) 289–307.

[29] G. Song, K. Zhang, Visual XML schemas based on reserved graph grammars, in: Proc. Int. Conf. Inf. Technol. Coding and Computing, 2004: pp. 687-691.

[30] K. Zhang, D.-Q. Zhang, Y. Deng, A Visual Approach to XML Document Design and Transformation, Proc. IEEE Symp. Human-Centric Comput. Lang. Environ., 2001: pp.312–319.

[31] K.-B. Zhang, M.A. Orgun, K. Zhang, A prediction-based visual approach for cluster exploration and cluster validation by HOV3. Lec. Notes Comput. Sci. 4702 (2007) 336-349.

[32] C. Zhao, J. Kong, K. Zhang, Design pattern evolution and verification using graph transformation, Proc. 40th Annual Hawaii International Conference on System Sciences (HICSS'2007), 2007: pp.290a-290a.

[33] C. Zhao, J. Kong, J. Dong, K. Zhang, Pattern-based Design Evolution Using Graph Transformation, J. Vis. Lang. Comput. 18 (2007) 378–398.

[34] C. Zhao, J. Kong, K. Zhang, Program behavior discovery and verification: A graph grammar approach, IEEE Trans. Softw. Eng. 36 (2010) 431–448.

[35] K. Zhang, J. Kong, Exploring semantic roles of Web interface components, Proc. Int. Conf. Mach. Web Intell., 2010: pp.8-14.

[36] K. Zhang, J. Kong, M. Qiu, G. Song, Multimedia layout adaptation through grammatical specifications, Multimedia Syst. 10 (2005) 245-260.

[37] J. Kong, K.-L. Ates, K. Zhang, Y. Gu, Adaptive mobile interfaces through grammar induction, Proc. Int. Conf. Tools with Artif. Intell. ICTAI, 2008: pp.133–140.

Spider Diagrams with Absence: Inference Rules for Clutter Reduction

Gem Stapleton[1] and Lopamudra Choudhury[2] and Mihir Chakraborty[2]
[1] Centre for Secure, Intelligent and Usable Systems,
University of Brighton, UK
[2] Jadavpur University, India
g.e.stapleton@brighton.ac.uk, choudhuryl@yahoo.com, mihirc4@gmail.com

Abstract

Spider diagrams represent sets, their cardinalities and, sometimes, the specific individuals within those sets. They are expressively equivalent to monadic first-order logic with equality. Typically, diagrammatic logics with this level of expressiveness are not equipped to directly *express the* absence *of an individual from a set. Instead, individuals must be asserted to be* present *and, thus, absent from the set's complement. The first time that absence could be directly asserted was in Venn-i. Since then, it been shown that in a related system called Venn-i^e (a monadic first-order logic without equality) the inclusion of absence information can significantly reduce diagram clutter. In this paper, we explore an extension of spider diagrams to include direct representation of the absence of individuals from sets. We identify necessary and sufficient conditions for satisfiability, allowing us to define an inconsistency rule allowing significant reductions in diagram clutter. Building on that, we introduce sound inference rules specifically related to spiders (which represent elements, individuals or their absence) that alter the levels of clutter in consistent diagrams. In the context of these rules, we explore the implications of including absence information for reducing clutter. In particular, we show that the significant benefits, in terms of clutter reduction, seen through the use of absence in Venn-i^e do not manifest to such an extent in spider diagrams.*

1. Introduction

The ability to negate statements plays a crucial role in all logics. The notion of absence is closely related to that of negation: $a \notin P$ (i.e. the individual a is not in the set P) indicates, informally speaking, that the individual a is *absent* from P. Indeed, the importance of negation should not be underestimated, "The capacity to negate is the capacity to refuse, to contradict, to lie, to speak ironically, to distinguish truth from falsity – in short, the capacity to be human" [7]. In diagrams research, though, it has long been believed that diagrams are not well equipped to make negated statements directly. Indeed, even simple statements like $a \notin P$ cannot be made explicitly in most Euler diagram-based logics, such as [10, 14, 18, 19]. Instead, these types of diagrams tend to assert $a \in \overline{P}$ (the complement of P).

DOI reference number: 10.18293/VLSS2018-032.

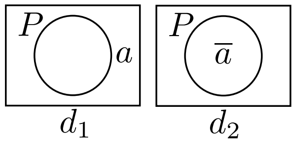


Figure 1. Asserting presence and absence.

There is an exception to this: Choudhury and Chakraborty developed a diagrammatic logic named Venn-i that allows $a \notin P$ to be *directly* expressed [4]. The Venn-i logic builds on Shin's Venn-I system [15], which exploits Peirce's $\otimes$-sequences to indicate the non-emptiness of sets [13]. Venn-i also uses i-sequences and $\bar{i}$-sequences to represent individuals and, respectively, their absence. Choudhury and Chakraborty adopt a classical interpretation, meaning that the absence of an individual from one set implies its presence in the complement[1]. An inspiration for Choudhury's and Chakraborty's work came from the notion of *abhāva* (absence). Abhāva, an important feature of ancient Indian knowledge systems, allocates a first class status to the absence of individuals.

Examples can be seen in figure 1. The diagram d_1 directly expresses that a is in $\overline{P}$, since the location of the symbol a is outside the curve P. From this, we can deduce that a is not in P, that is, a is absent from P. By contrast,

[1]A related system, developed by Bhattacharjee et al. focuses on a non-classical interpretation of absence [2]. In that system, the absence of an individual from one set *does not* imply its presence in the complement. They devised a sound and complete set of inference rules which allow diagrammatic proofs to be written when absence information is given.

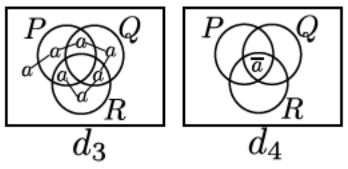


Figure 2. Clutter reduction.

the diagram d_2 directly expresses that a is absent from P by placing $\overline{a}$ inside P. Thus, the use of $^-$ depicts the absence of individuals from a set. Whilst this example is sufficient to illustrate the use of absence, it does not demonstrate the role that absence can play in reducing diagram clutter. We explore this more fully later in the paper. For now, suppose we want to express that a is absent from $P \cap Q \cap R$. Referring to figure 2, this is achieved indirectly by d_3 which expresses that $a \in \overline{P \cap Q \cap R}$ (i.e. the complement of $P \cap Q \cap R$). The diagram d_4 is semantically equivalent but instead expresses the required statement, $a \notin P \cap Q \cap R$, directly using absence. The diagram d_4 is arguably less cluttered than d_3.

Clutter in Euler diagrams was studied by John et al. [12]: they devised a theoretical measure of clutter. Alqadah et al. empirically found that increasing levels of clutter in Euler diagrams negatively impacts user task performance [1]. Whilst empirically studying concept diagrams [11] (which extend Euler diagrams with syntax for individuals amongst other things), Hou et al. found that the diagrams where people had trouble performing tasks were those with the higher levels of clutter [8, 9]. Hence, there is clearly a need to be able to measure the level clutter in diagrams generally and its impact on end-user task performance.

In previous work we, with Burton, demonstrated that explicitly representing the absence of individuals allows information to be presented in a less cluttered way [3]. This was in an Euler diagram system, called Venn-i^e, that incorporated $\otimes$-sequences, i-sequences (like the a-sequence in d_3) and $\overline{i}$-sequences (like $\overline{a}$ in d_4). We note here that, just as in Venn-i, distinct i-sequences need not represent distinct individuals. As such, Venn-i^e is a monadic first-order logic (without equality). An empirical evaluation suggested, for Venn-i^e, high levels of clutter arising from individuals is detrimental to human cognition [16].

There are two key points: (a) rising levels of diagram clutter negatively impacts human cognition and (b) representing absence directly in systems with the expressiveness of monadic first-order logic (without equality) can bring reductions in clutter. An obvious question arises: does the inclusion of absence in more expressive systems still bring with it possibility of reducing clutter to the same extent? This paper helps us to understand clutter arising from spiders (akin to sequences) in an extended version of spider diagrams, which we call enhanced spider diagrams. This increases the level of expressiveness, over which we can explore the role of absence, to monadic first-order logic with equality. In particular, we make the following contributions:

- identify how to extend spider diagrams to include absence information (section 2).
- formalize the syntax and semantics of enhanced spider diagrams (section 3),
- identify necessary and sufficient conditions for enhanced spider diagrams to be unsatisfiable (section 4),
- introduce inference rules for enhanced spider diagrams that apply specifically to spiders (section 5),
- discuss how the inference rules can be used to alter the level of clutter present in enhanced spider diagrams (section 6), and
- draw comparison to the Venn-i^e case in section 7.

We conclude and identify future work in section 8. This paper directly extends [17].

2. Representing Absence in Spider Diagrams

We now proceed to, briefly, show how absence can be incorporated into spider diagrams [5], which typically include *existential* spiders for denoting the existence of elements in sets. In addition, they have been studied with the inclusion of *constant spiders* [18] which, in this paper, we refer to as *positive spiders*. These spiders represent the presence of specific individuals in particular sets.

An example can be seen in figure 3. The spider diagram d_5 expresses the following: (i) due to the spatial relationships between the curves, R is disjoint from P and from Q; (ii) due to the inclusion of two spiders, there are at least two elements, one of which (denoted by the existential spider comprising two nodes) is in P and the other of which (denoted by the positive spider a) is the individual a and is in the set $P \backslash Q$, and (iii) due to the shading, combined with the existential spider, there is at most one element in $P \cap Q$.

The diagram d_6 augments d_5 with additional information expressed by four *negative* spiders: (iv) the individual b is not in $\overline{P}$ and the individual c is not in R. Therefore, from (iv), b is in the set P. An obvious question then arises: should the individual b be necessarily different from the two elements represented by the existential spider and the positive spider a? In our view, the most *diagrammatic* interpretation is that b is *not* necessarily different. It seems natural to say that two existential or positive spiders placed in a common region represent distinct individuals since they are represented by distinct syntactic devices. However, it does not seem very diagrammatic for d_6 to force P to contain at least *three* elements when we can see only *two* spiders inside P. This observation suggests that the diagram would not be *well-matched* to its semantics [6] if we forced b to denote the presence of an *additional* element in P. Therefore, in our extension of spider diagrams to include absence information directly, via *negative* spiders, we do not interpret negative spiders as providing distinctness information about individuals, unlike existential and positive spiders.

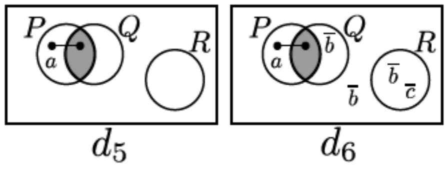


Figure 3. Incorporating absence.

3. Syntax and Semantics

Having introduced how absence can be incorporated into spider diagrams, we now formally define the syntax and semantics of the enhanced system. It is helpful for us to have a countably infinite set of labels from which all labels used on the curves in any diagram are drawn; we call this set $\mathcal{L}$. A **zone** is a pair, (in, out), where in and out are finite, disjoint subsets of $\mathcal{L}$. Given $\mathcal{L}$, the set of all zones is denoted $\mathcal{Z}$. In figure 3, each diagram has five zones, such as the one inside both P and Q but outside R; this zone is $(\{P,Q\},\{R\})$. We also have a countably infinite set of constant symbols, denoted $\mathcal{C}$, which are used as labels for *positive* and *negative* spiders. So, in figure 3, spider labels a, b and c appear. Using these predefined sets, we can now formally define an *enhanced spider diagram* at the abstract syntax level:

Definition 1. *An* ***enhanced spider diagram***, *d, is a tuple, $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ such that:*

1. *L is a finite set of labels chosen from $\mathcal{L}$.*
2. *Z is a set of zones where $(\emptyset, L) \in Z$ and for all (in, out) in Z, $in \cup out = L$.*
3. *ShZ is a subset of Z whose elements are called* ***shaded zones****.*
4. *ES, PS, NS are finite pairwise disjoint sets whose elements are called* ***existential spiders, positive spiders,*** *and* ***negative spiders*** *respectively.*
5. *η: $ES \cup PS \cup NS \to \mathbb{P}Z \backslash \{\emptyset\}$ returns the location of each spider.*
6. *ρ: $PS \cup NS \to \mathcal{C}$ returns the label of each positive and negative spider.*

We further define $S(d) = ES \cup PS \cup NS$ to be the set of ***spiders*** *in d.*

In figure 3, d_6 has the following abstract syntax:

1. label set $L = \{P, Q, R\}$,
2. zone set $Z = \{(\{P\},\{Q,R\}), (\{Q\},\{P,R\}), (\{P,Q\},\{R\}), (\{R\},\{P,Q\}), (\emptyset,\{P,Q,R\})\}$,
3. shaded zone set $ShZ = \{(\{P,Q\},\{R\})\}$,
4. existential spider set $ES = \{\sigma_1\}$, positive spider set $PS = \{\sigma_a\}$, and negative spider set $NS = \{\sigma_{b,1}, \sigma_{b,2}, \sigma_{b,3}, \sigma_c\}$,
5. spider locations given by

$$\begin{aligned}
\eta(\sigma_1) &= \{(\{P\},\{Q,R\}), (\{P,Q\},\{R\})\},\\
\eta(\sigma_a) &= \{(\{P\},\{Q,R\}),\\
\eta(\sigma_{b,1}) &= \{(\{Q\},\{P,R\})\},\\
\eta(\sigma_{b,2}) &= \{(\emptyset,\{P,Q,R\})\},\\
\eta(\sigma_{b,3}) &= \{(\{R\},\{P,Q\})\},\\
\eta(\sigma_c) &= \{(\{R\},\{P,Q\})\},
\end{aligned}$$

   and
6. the labels of the positive and negative spiders are given by $\rho(\sigma_a) = a$, $\rho(\sigma_{b,1}) = \rho(\sigma_{b,2}) = \rho(\sigma_{b,3}) = b$, and $\rho(\sigma_c) = c$.

We now proceed to define some useful syntactic notions. For example, in figure 3, the zone $(\{P,R\},\{Q\})$ is *missing* from d_6 (and d_5). Since d_6 is taken to assert that R is disjoint from both P and Q, the set $P \cap R \cap \overline{Q}$ represented by this zone must be empty.

Definition 2. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. The* ***missing zones*** *of d are elements of*

$$MZ(d) = \{(in, out) \in \mathcal{Z} : in \cup out = L\} \backslash Z.$$

So, d_6 has three missing zones, giving

$$MZ(d_6) = \{(\{P,R\},\{Q\}), (\{Q,R\},\{P\}), (\{P,Q,R\},\emptyset)\}.$$

Missing zones need not be the only zones that represent empty sets. In particular, shading placed in zones enforces an upper bound on set cardinality: in a shaded region, all elements must be represented by existential or positive spiders. So, a shaded region containing no part of any such spider represents the empty set. There are no *necessarily* empty zones in d_6 but, in figure 4, the diagram d_7 only contains empty zones (all zones are shaded and there are no existential or positive spiders).

Definition 3. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. The* ***empty zones*** *of d are elements of*

$$EZ(d) = \{z \in ShZ : \forall \sigma \in ES \cup PS \quad z \notin \eta(\sigma)\}.$$

It is also useful to identify zones that represent sets in which an individual cannot lie, due to the information provided by negative spiders. For instance, in d_6 of figure 3, b is not in (the sets represented by) the three zones which contain $\overline{b}$.

Definition 4. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. Let c be a constant from $\mathcal{C}$. The* ***negative zones*** *for c in d are elements of $NZ(c,d)$ where*

$$NZ(c,d) = \{z \in Z : \exists \sigma \in NS \quad \eta(\sigma) = \{z\} \wedge \rho(\sigma) = c\}.$$

So, in d_6, we have:

- no negative zone for a: $NZ(a, d_6) = \emptyset$,
- three negative zones for b since there are three $\overline{b}$s placed in single zones: $NZ(b, d_6) = \{(\{Q\}, \{P, R\}), (\emptyset, \{P, Q, R\}), (\{P\}, \{P, Q\})\}$ and
- one negative zone for c, since there is one $\overline{c}$ placed in a single zone: $NZ(c, d_6) = \{(\{R\}, \{P, Q\})\}$.

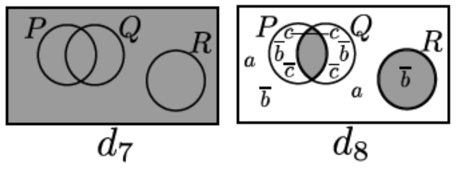


Figure 4. Inconsistency and clutter.

Our attention now turns to semantics. As is standard, we interpret the curve labels as subset of some (non-empty) universal set and constant symbols as elements.

Definition 5. *An* ***interpretation***, *$\mathcal{I}$, is a triple, $\mathcal{I} = (U, \psi, \Psi)$, such that*

1. *U is a non-empty set, called the* ***universal set***,
2. *$\psi: \mathcal{C} \to U$ maps constants to elements in U, and*
3. *$\Psi: \mathcal{L} \to \mathbb{P}U$ maps curve labels to subsets of U.*

The function Ψ is extended to interpret zones and sets of zones (regions) as follows: for each zone, (in, out),

$$\Psi(z) = \bigcap_{l \in in} \Psi(l) \cap \bigcap_{l \in out} (U \backslash \Psi(l))$$

and for each region, r, $\Psi(r) = \bigcup_{z \in r} \Psi(z)$.

Given an interpretation, it can either agree with the intended intuitive meaning of a diagram or not. For instance, in figure 3, any interpretation where $\Psi(P) \cap \Psi(R) \neq \emptyset$ does not agree with the intended intuitive meaning of d_6: because the curves P and R do not overlap, d_6 tells us that $\Psi(P) \cap \Psi(R) = \emptyset$. The semantics of diagrams are given by the set of interpretations that match our expectations of what the diagram expresses. We therefore give precise conditions under which an interpretation *satisfies* an enhanced spider diagram; satisfying interpretations are called *models*.

Definition 6. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram and let $\mathcal{I} = (U, \psi, \Psi)$ be an interpretation. Then $\mathcal{I}$ is a* ***model*** *for d provided there exists a function, $\psi': ES \cup PS \cup NS \to U$, where, for each positive and negative spider, σ, in $PS \cup NS$ we have $\psi'(\sigma) = \psi(\rho(c))$ and the following conditions hold.*

1. ***Missing Zones Condition:*** *missing zones represent empty sets, that is for all $z \in MZ(d)$, $\Psi(z) = \emptyset$.*
2. ***Shaded Zones Condition:*** *shaded zones represent sets containing only elements represented by existential or positive spiders, that is for all $z \in ShZ$,*
$$\Psi(z) \subseteq \{\psi'(\sigma) : \sigma \in ES \cup PS\}.$$
3. ***Spider Distinctness Condition*** *no two existential or positive spiders represent the same element, that is for all σ_1 and σ_2 in $ES \cup PS$,*
$$\psi'(\sigma_1) = \psi'(\sigma_2) \Rightarrow \sigma_1 = \sigma_2.$$
4. ***Existential Spiders Condition*** *each existential spider represents an element in the region in which it is placed, that is for all σ in ES, $\psi'(\sigma) \in \Psi(\eta(\sigma))$.*
5. ***Positive Spiders Condition*** *each positive spider represents an element in the region in which it is placed, that is for all σ in PS, $\psi'(\sigma) \in \Psi(\eta(\sigma))$.*
6. ***Negative Spiders Condition*** *each negative spider represents an element that is not some zone in the region in which it is placed, that is for all σ in NS, there exists z in $\eta(\sigma)$ where $\psi'(\sigma) \notin \Psi(z)$.*

Such a function ψ' that makes the above conditions true for d is called ***valid***. *If $\mathcal{I}$ models d then $\mathcal{I}$* ***satisfies*** *d. Diagrams with no models are* ***unsatisfiable***.

The interpretation with $U = \{1, 2, 3, 4\}$, $\psi(a) = 1$, $\psi(b) = 2$, $\psi(c) = 3$, $\Psi(P) = \{1, 2\}$, $\Psi(Q) = \{2, 3\}$ and $\Psi(R) = \{4\}$ is a model for d_6. However, if instead $\Psi(Q) = \{1, 2, 3\}$ then the resulting interpretation fails to model d_6; for instance, $\psi(a) = 1$ but, since the positive spider labelled a is in the zone $(\{P\}, \{Q, R\})$ and we have

$$\begin{aligned}\Psi(\{P\}, \{Q, R\}) &= \Psi(P) \cap (U \backslash \Psi(Q)) \cap (U \backslash \Psi(R)) \\ &= \emptyset,\end{aligned}$$

the positive spiders condition fails under any ψ' that agrees with ψ.

One important, yet straightforward to verify, insight is that distinct zones in a diagram represent disjoint sets. In essence, given two such zones (in_1, out_1) and (in_2, out_2), we know that $in_1 \cup out_1 = in_2 \cup out_2$. Therefore, there is a label, l say in $in_1 \cap out_2$ or in $in_2 \cap out_1$. So one zone is a subset of $\Psi(l)$ and the other is a subset of $U \backslash \Psi(l)$. This leads us to lemma 1:

Lemma 1. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. In all models, $I = (U, \psi, \Psi)$, for d,*

$$\Psi(z_1) \cap \Psi(z_2)$$

for all zones z_1 and z_2 in $Z \cup MZ$. That is, distinct zones in d or which are missing from d represent disjoint sets.

We now present two results relating to, respectively, empty zones and negative zones. Firstly, we establish that empty zones represent empty sets in models:

Lemma 2. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. In all models, $I = (U, \psi, \Psi)$, for d, $\Psi(EZ(d)) = \emptyset$.*

Proof. Suppose $I = (U, \psi, \Psi)$ is a model for d. Let $z \in EZ(d)$. We show $\Psi(z) = \emptyset$. Since I is a model for d, there exists a valid $\psi' \colon ES \cup PS \cup NS \to U$ for d. Choose such a ψ'. Since z is in $EZ(d)$ we know that z is shaded. So, by the shaded zones condition,

$$\Psi(z) \subseteq \{\psi'(\sigma) : \sigma \in ES \cup PS\}.$$

Let σ be an existential or positive spider in d, so $\sigma \in ES \cup PS$. Then, by the existential and positive spiders conditions for d we know

$$\psi'(\sigma) \in \Psi(\eta(\sigma)).$$

Now, since z is in $EZ(d)$, we know that $z \notin \eta(\sigma)$. Since distinct zones in d represent disjoint sets (lemma 1), we deduce that

$$\Psi(z) \cap \Psi(\eta(\sigma)) = \emptyset.$$

Therefore,

$$\psi'(\sigma) \notin \Psi(z).$$

Since σ was an arbitrary existential or positive spider and the shaded zones condition holds (i.e. $\Psi(z)$ only contains elements represented by existential or positive spiders), we deduce that

$$\Psi(z) = \emptyset.$$

Hence $\Psi(EZ(d)) = \emptyset$ as required. □

Secondly, we show that, for any given spider label, c, its associated negative zones do not contain the individual represented by c. For instance, in d_6 we already saw that $NZ(b, d_6)$ includes $(\{Q\}, \{P, R\})$. This zone is the location for a negative spider, $\sigma_{b,1}$, labelled b, that is $\eta(\sigma_{b,1}) = \{(\{Q\}, \{P, R\})\}$. In any model for d_6, the negative spiders condition tells us that $\psi'(b) = \psi(b) \notin \Psi(\{Q\}, \{P, R\})$. Importantly, the negative zones arise precisely from the negative spiders whose locations are single zones, from which the proof of the following lemma readily follows:

Lemma 3. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. In all models, $I = (U, \psi, \Psi)$, for d, for all $c \in \mathcal{C}$, it is the case that $\psi(c) \notin \Psi(NZ(c, d))$.*

Proof. Suppose $I = (U, \psi, \Psi)$ is a model for d and let $c \in \mathcal{C}$. If $NZ(c, d) = \emptyset$ then $\Psi(NZ(c, d)) = \emptyset$ and we have $\psi(c) \notin \Psi(NZ(c, d))$. Alternatively, let $z \in NZ(c, d)$. We show $\psi(c) \notin \Psi(z)$. Since I is a model for d, there exists a valid $\psi' \colon ES \cup PS \cup NS \to U$ for d. Choose such a ψ'. Now, since $z \in NZ(c, d)$, we know that there exists a negative spider, σ, in d with label c (i.e. $\rho(\sigma) = c$) and whose location is z (i.e. $\eta(\sigma) = \{z\}$). By the negative spiders condition for d, we know that

$$\psi'(\sigma) \notin \Psi(z),$$

as required. Hence $\psi(c) \notin \Psi(NZ(c, d))$. □

Thus, the definitions of empty zones and negative zones have the expected properties in models. One further result will be useful to us, which establishes that in a model for diagram, d, each element in the universal set, U, must lie in some zone in d.

Lemma 4. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. In all models, $I = (U, \psi, \Psi)$, for d,*

$$\bigcup_{z \in Z} \Psi(z) = \Psi(Z) = U.$$

4 Inconsistency and Satisfiability

A key motivation for this work is to explore the role of absence in clutter reduction. We begin by observing that every unsatisfiable diagram is semantically equivalent to a diagram containing no spiders. For example, in figure 4, d_7 is inconsistent: every interpretation has a non-empty universal set yet, since d_7 is entirely shaded and contains no spiders, the shading and missing zones conditions can never both be satisfied (a non-empty universal set implies at least one zone represents a non-empty set). The diagram d_8 is also unsatisfiable, for any one of the following reasons:

1. There are two positive spiders both with the same label, a, meaning that the spider distinctness condition can never hold.
2. The negative $\overline{b}$ spiders together imply that b must lie in $P \cap Q \cap \overline{R}$, yet this region is entirely shaded and contains no part of an existential or positive spider. Therefore, d_8 implies two contradictory statements: $b \in P \cap Q \cap \overline{R}$ and $P \cap Q \cap \overline{R} = \emptyset$.
3. The negative $\overline{c}$ spiders tell us that c cannot lie in $P \cap \overline{Q} \cap \overline{R}$ or in $Q \cap \overline{P} \cap \overline{R}$, yet the positive spider c expresses $c \in (P \cap \overline{Q} \cap \overline{R}) \cup (Q \cap \overline{P} \cap \overline{R})$. Clearly both these assertions cannot be true at the same time.

Since d_8 is inconsistent, it is semantically equivalent to d_7, which is visually less cluttered. Since every inconsistent diagram is semantically equivalent to a diagram that is entirely shaded and contains no spiders, identifying necessary and sufficient conditions for unsatisfiability provides some insight into how negative spiders can lead to clutter reduction.

One important feature of the last example was that it was not possible to find zones that represent sets containing certain individuals. For instance, there was no zone for the individual c since it was taken to be present in $(\{P\}, \{Q, R\})$ or $(\{Q\}, \{P, R\})$ yet absent from both $(\{P\}, \{Q, R\})$ and $(\{Q\}, \{P, R\})$. For a diagram to be satisfiable, for each constant, c_i, we must be able to select a zone that, in some model, represents a set containing the individual represented by c_i.

Definition 7. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. A **zone selection function** for d is a mapping, $f\colon ES \cup PS \cup NS \to Z$ which ensures the following hold:*

1. *the zone selected for each existential and positive spider is one of the zones in its location: for all $\sigma \in ES \cup PS$, $f(\sigma) \in \eta(\sigma)$,*
2. *the zone selected for a negative spider cannot be a negative zone for its label: for all $\sigma \in NS$, $f(\sigma) \notin NZ(\rho(\sigma), d)$ and*
3. *if a shaded zone is selected for a negative spider then it must also be selected for an existential or positive spider: for all $\sigma \in NS$, if $f(\sigma) \in ShZ$ then there exists $\sigma' \in ES \cup PS$ such that $f(\sigma') = f(\sigma)$, and*
4. *spiders with the same label have the same zone selected: for all σ_1, σ_2 in $PS \cup NS$ if $\rho(\sigma_1) = \rho(\sigma_2)$ then $f(\sigma_1) = f(\sigma_2)$.*

The zone selection function identifies, for each spider, a specific zone in the diagram, d. Any given zone selection function can be used to define a model for d, where the individuals represented by the spiders are in the sets represented by the selected zones. For the purposes of intuition, we consider each of the conditions of definition 7. Condition 1 arises from the need for each existential and positive spider to represent an element in (the set represented by) one of the zones of its location. Condition 2 captures the fact that negative zones cannot contain, in a model for d, the individual represented by $\rho(\sigma)$. Condition 3 considers the interaction between negative spiders and shading. The zone selected for a negative spider, if shaded, cannot represent the empty set in a model. This is enforced by the requirement that some existential or positive spider has been assigned to that shaded zone. The last condition requires the same zone to be selected for spiders with a common label because such spiders represent the same individual.

Using d_6 in figure 3 as an example, adopting the previously given abstract syntax, we can define

$$\begin{aligned} f(\sigma_1) &= \{(\{P, Q\}, \{R\})\}, \\ f(\sigma_a) &= \{(\{P\}, \{Q, R\})\}, \\ f(\sigma_{b,1}) &= \{(\{P, Q\}, \{R\})\} \\ f(\sigma_{b,2}) &= \{(\{P, Q\}, \{R\})\} \\ f(\sigma_{b,3}) &= \{(\{P, Q\}, \{R\})\}, \\ f(\sigma_c) &= \{(\{P, Q\}, \{R\})\}. \end{aligned}$$

Under this zone selection function, a model can be generated for d_6 where b and c represent the same individual and that individual is in the set represented by the zone $\{(\{P, Q\}, \{R\})\}$. Under any valid ψ' that respects f, this individual is also represented by σ_1, due to the presence of shading.

We are now in a position to define the notion of (in)consistency:

Definition 8. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. Whenever the following conditions all hold d is **consistent**:*

1. *If all of the zones in Z are shaded then there is at least one existential or positive spider.*
2. *No two positive spiders have the same label, that is, the function ρ is injective when its domain is restricted to PS.*
3. *There exists a zone selection function, f, for d.*

*If d is not consistent then d is **inconsistent**.*

So, d_6 in figure 3 is consistent whereas d_7 and d_8 in figure 4 are inconsistent.

In order to prove that a consistent diagram, d is satisfiable, our approach is to construct an interpretation that is a model for d. To do so, we make use of a foot selection function, which we know exists since d is consistent. The first part of this task is to construct a suitable universal set. Now, for each existential spider and positive spider in d, we require the existence of a distinct element in U. So, our strategy is to include all of the existential spiders and the positive spiders in U. However, this alone does not ensure enough elements will be in U for all of the spiders. In particular, negative spiders may require the existence of elements in non-shaded zones where there is no element arising from an existential or positive spider. Thus, our universal set will also include, as elements, all of the non-shaded zones. In fact, we will establish that this provides sufficient elements in U.

Once U is constructed, we must build a suitable mapping from constant symbols to elements in U. For the constants that are used to label positive spiders this is straightforward: we map the label to the (unique) positive spider in d (and, thus, in U) with that label. Things are less straightforward for a constant, c, that does not label a positive spider but does appear on one or more negative spiders. As indicated above, if the selected zone for such spiders is non-shaded then we simply map the associated constant symbol to that zone. What, then, if the selected zone is shaded? Here, we focus on the third requirement of a foot selection function: if a shaded zone is selected for a negative spider then it must also be selected for a positive or existential spider. This requirement arises since, in shaded zones, all elements must be represented by existential or positive spiders. So, to map a constant symbol, c, appearing in d on negative spiders only, whose selected zone is shaded, we must choose one of these elements of U (i.e an existential or positive spider with the same selected zone) to be the interpretation of c. Our next definition allows us to pair such negative spiders with existential or positive spiders:

Definition 9. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram with foot selection function f. A* ***selected negative spider mapping*** *for d given f is a function, $s\colon NS' \to ES \cup PS$ which ensures*

$$s(\sigma) = \sigma' \Rightarrow f(\sigma) = f(\sigma')$$

where

$$NS' = \{\sigma \in NS : \neg\exists\sigma' \in PS \rho(\sigma) = \rho(\sigma') \wedge f(\sigma) \in ShZ\}.$$

Lemma 5. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram with foot selection function f. Then d, given f, has a selected negative spider mapping.*

Proof Sketch. The proof follows trivially from condition 3 of definition 7. □

Definition 10. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram with a zone selection function, f, and selected negative spider mapping, s. A* ***standard interpretation****, $I = (U, \psi, \Psi)$, for d given f and s is defined as follows.*

1. *the universal set comprises the non-shaded zones, the existential spiders and the positive spiders:*

$$U = (Z \backslash ShZ) \cup ES \cup PS$$

2. *$\psi\colon \mathcal{C} \to U$ is defined as follows, where u is an arbitrary element in U:*

$$\psi(c) = \begin{cases} \sigma & \text{when } \sigma \text{ is in } PS \text{ with } \rho(\sigma) = c \\ f(\sigma) & \sigma \text{ is in } NS\text{, no positive spider has} \\ & \text{label } c\text{, } \rho(\sigma) = c \text{ and } f(\sigma) \in Z\backslash ShZ \\ s(\sigma) & \sigma \text{ is in } NS' \text{ with } \rho(\sigma) = c \\ u & \text{otherwise.} \end{cases}$$

3. *$\Psi\colon \mathcal{L} \to \mathbb{P}U$ is defined as follows:*

$$\begin{aligned} \Psi(l) \quad = \quad & \{\sigma \in U\backslash Z : f(\sigma) = (in, out) \wedge l \in in\} \cup \\ & \{(in, out) \in U \cap Z : l \in in\}. \end{aligned}$$

Lemma 6. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram with a zone selection function, f, and selected negative spider mapping, s. Let $I = (U, \psi, \Psi)$ be a standard interpretation for d given f and s. Let $z \in Z \cup MZ$. Then*

$$\Psi(z) = \{\sigma \in U\backslash Z : f(\sigma) = z\} \cup (U \cap \{z\}).$$

Theorem 1. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram with a zone selection function, f, and selected negative spider mapping, s. Let $I = (U, \psi, \Psi)$ be a standard interpretation for d given f and s. Then I is a model for d.*

Proof. We start by showing that U is not empty, since this is a requirement for I to be an interpretation. Trivially, if there is a non-shaded zone then U is non-empty. Else, all zones are shaded. Since d is consistent, this implies that there is at least one existential or positive spider in d. By construction, such a spider is in U. In both cases, U is non-empty as required.

We now show that the conditions for U to be a model for d are met. We start by defining a function

$$\psi'\colon ES \cup PS \cup NS \to U$$

by

$$\psi'(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in ES \cup PS \\ \sigma' & \text{if } \sigma \in NS \text{ and} \\ & \text{there is a } \sigma' \text{ in } PS \text{ where } \rho(\sigma) = \rho(\sigma') \\ f(\sigma) & \sigma \text{ is in } NS\text{, no positive spider has} \\ & \text{label } \rho(\sigma)\text{, and } f(\sigma) \in Z\backslash ShZ \\ s(\sigma) & \sigma \text{ is in } NS'. \end{cases}$$

Next, we show that ψ' is valid. Firstly, consider the missing zones condition. Let z be a missing zone. By lemma 6, we know that

$$\Psi(z) = \{\sigma \in U\backslash Z : f(\sigma) = z\} \cup (U \cap \{z\}).$$

Since z is missing, z is not in U, so

$$\Psi(z) = \{\sigma \in U\backslash Z : f(\sigma) = z\}.$$

But the zone selection function, f, has codomain Z, so no spider maps to z under f. Therefore $\Psi(z) = \emptyset$ as required. Hence the missing zones condition holds.

Next we consider the shaded zones condition. Let z be a shaded zone. By lemma 6, we know that

$$\Psi(z) = \{\sigma \in U\backslash Z : f(\sigma) = z\} \cup (U \cap \{z\}).$$

Since z is shaded, z is not in U, so

$$\Psi(z) = \{\sigma \in U\backslash Z : f(\sigma) = z\}.$$

From this it follows that

$$\Psi(z) \subseteq ES \cup PS$$

and, noting that ψ' maps existential and positive spiders to themselves (i.e. over the restricted domain $ES \cup PS$, ψ' is the identity function) we have

$$\Psi(z) \subseteq \{\psi'(\sigma) : \sigma \in ES \cup PS\}$$

as required. Hence the shaded zones condition holds.

For the spider distinctness condition, we note that ψ' is injective when its domain is restricted to $ES \cup PS$. Hence the spider distinctness condition holds. There are three remaining conditions to verify, all related to spiders. Let σ be an existential spider. Consider $\Psi(f(\sigma))$. We know

$$\Psi(f(\sigma)) = \{\sigma' \in U \backslash Z : f(\sigma') = f(\sigma)\} \cup (U \cap \{f(\sigma)\})$$

by lemma 6. Clearly, σ is in $\Psi(f(\sigma))$ and, since $\sigma = \psi'(\sigma)$, it follows that

$$\psi'(\sigma) \in \Psi(f(\sigma)).$$

Moreover, since f is a zone selection function, $f(\sigma)$ is in $\eta(\sigma)$. We deduce that

$$\psi'(\sigma) \in \Psi(f(\sigma)) \subseteq \Psi(\eta(\sigma)).$$

Therefore the existential spiders condition holds. The positive spiders condition similarly holds.

All that remains is to show that the negative spiders condition holds. Let σ be a negative spider. If $\eta(\sigma)$ contains at least two zones then the negative spiders condition trivially holds: $\psi'(\sigma)$ can be in at most one of the sets represented by the two or more zones. We can assume, then, in what follows that $|\eta(\sigma)| = 1$. There are three cases to consider. In all cases, we aim to show that $\psi'(\sigma) \notin \Psi(z)$ where $\eta(\sigma) = \{z\}$.

Case (1): $\sigma \in NS$ and there is a σ' in PS where $\rho(\sigma) = \rho(\sigma')$. Here, $\psi'(\sigma) = \psi'(\sigma')$ by definition and, by the reasoning above we know that

$$\psi'(\sigma) = \psi'(\sigma') \in \Psi(f(\sigma')).$$

Moreover, since f is a foot selection function and $\rho(\sigma) = \rho(\sigma')$ we also know that $f(\sigma) = f(\sigma')$. Thus,

$$\psi'(\sigma) = \psi'(\sigma') \in \Psi(f(\sigma')) = \Psi(f(\sigma)).$$

Since $\eta(\sigma)$ contains a single zone, z, by assumption, it follows that $z \in NZ(\rho(\sigma), d)$. Again, since f is a zone selection function, we see that

$$f(\sigma) \notin NZ(\rho(\sigma), d)$$

and we can deduce that $f(\sigma) \neq z$. Hence $f(\sigma) \notin \eta(\sigma)$, so $\psi'(\sigma) \notin \Psi(\eta(\sigma))$ as required.

Case (2): σ is in NS, no positive spider has label $\rho(\sigma)$, and $f(\sigma) \in Z \backslash ShZ$. In this case, $\psi'(\sigma) = f(\sigma)$, by definition. We know, by lemma 6, that

$$\Psi(f(\sigma)) = \{\sigma' \in U \backslash Z : f(\sigma') = f(\sigma)\} \cup (U \cap \{f(\sigma)\}).$$

Since U includes all non-shaded zones, and $f(\sigma)$ is such a zone, we see that

$$\psi'(\sigma) = f(\sigma) \in \Psi(f(\sigma))$$

As f is a foot selection function, $f(\sigma)$ is not in $NZ(\rho(\sigma), d)$. Moreover, because $\eta(\sigma)$ contains only one zone, z, (by assumption), we know that z is in $NZ(\rho(\sigma), d)$. Now, since distinct zones in d represent disjoint sets, we have

$$\psi'(\sigma) \notin \Psi(z) = \Psi(\eta(\sigma))$$

as required.

Case (3): σ is in NS'. Here, we use the fact that $s(\sigma) = \sigma'$, for some σ' in $ES \cup PS$, where $f(\sigma) = f(\sigma')$. We already saw that

$$\psi'(\sigma') \in \Psi(f(\sigma')) = \Psi(f(\sigma)).$$

Now, $\psi'(\sigma') = \sigma'$, by construction, and $\psi'(\sigma) = \sigma'$, so

$$\psi'(\sigma) \in \Psi(f(\sigma)).$$

Again, as the location, $\eta(\sigma)$, contains a single zone and $f(\sigma)$ is not a negative zone for $\rho(\sigma)$ in d, it follows that

$$\psi'(\sigma) \notin \Psi(\eta(\sigma)).$$

Hence, in all cases the negative spiders condition holds and we have shown that I is a model for d. □

We can now state and prove the main result of this section:

Theorem 2. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. Then d is consistent if and only if d is satisfiable.*

Proof. For the first part of the proof, assume d is consistent. Then, by theorem 1, a standard model satisfies d. Hence, d is satisfiable.

For the converse, assume that d is satisfiable. We must show that d is consistent. Let $I = (U, \psi, \Psi)$ be a model for d and assume ψ': $ES \cup PS \cup NS \to U$ is a valid mapping of spiders to universal set elements. We now consider three cases, relating to the conditions for d to be consistent.

Firstly, then, assume that d is entirely shaded. Our task is to show that d contains an existential or positive spider. Since I is an interpretation, U is not empty. Let e be an element of U. Then there is some zone, z, in d where $e \in \Psi(z)$, by lemma 4. By assumption, z is shaded so

$$\Psi(z) \subseteq \{\psi(\sigma) : \sigma \in ES \cup PS\}$$

by the shaded zones condition. Therefore

$$e \in \{\psi(\sigma) : \sigma \in ES \cup PS\}$$

from which it trivially follows that there is a spider in ES or PS as required.

Secondly, we must show that no two positive spiders have the same label. Suppose there are two positive spiders, σ_1 and σ_2, with the same label. By the spider distinctness condition, we know that

$$\psi'(\sigma_1) \neq \psi'(\sigma_2).$$

But since ψ' is valid, we also know that

$$\psi'(\sigma_1) = \psi(\rho(\sigma_1)) = \psi(\rho(\sigma_2)) = \psi'(\sigma_2)$$

which is a contradiction. Hence no two positive spiders have the same label.

Lastly, we must show that there is a zone selection function for d. We construct $f \colon ES \cup PS \cup NS \to Z$ and show that f is such a function. Consider, for each spider, σ, in $ES \cup PS \cup NS$, $\psi'(\sigma)$. Since $\psi'(\sigma)$ is in U, we know, by lemma 4 and since distinct zones in d represent disjoint sets, there is a unique zone, z, such that $\psi'(\sigma) \in \Psi(z)$. We define, for each σ in $ES \cup PS \cup NS$,

$$f(\sigma) = z$$

where $\psi'(\sigma) \in \Psi(z)$ and $z \in Z$. We must now verify that f satisfies the four conditions required of a zone selection function.

1. We must show that the zone selected for each existential and positive spider, σ, is a zone in its location, that is $f(\sigma) \in \eta(\sigma)$. By definition, $f(\sigma)$ is the zone whose interpretation under Ψ contains $\psi'(\sigma)$. By the existential and positive spiders conditions, we know that $\psi'(\sigma) \in \Psi(\eta(\sigma))$. So, we have both

$$\begin{array}{lcl} \psi'(\sigma) & \in & \Psi(f(\sigma)) \qquad \text{and} \\ \psi'(\sigma) & \in & \Psi(\eta(\sigma)) \end{array}$$

   Since distinct zones in d represent disjoint sets, it follows that $f(\sigma) \in \eta(\sigma)$ as required.

2. Now we must show that the zone, $f(\sigma)$, selected for a negative spider, σ is not in $NZ(\rho(\sigma), d)$. Let σ_1,..., σ_n be all of the negative spiders in d with the same label as σ. Then we know that $\psi'(\sigma_1) = ... = \psi'(\sigma_n)$, so all of these spiders have the same zone, z, selected under f. For each spider, either z is not in it's location or, by the negative spiders condition, there is another zone in $\eta(z)$. For z to be a negative zone for σ, it would have to be the case that z is the only zone in the location of one σ_1, ..., or σ_n. As we have just seen, this is not the case, so z is not negative for σ. That is $f(\sigma) \notin NZ(\rho(\sigma), d)$, as required.

3. Suppose that a shaded zone is selected by f for some negative spider, σ. We must show that $f(\sigma)$ is also selected for some existential or positive spider. By the shaded zones condition,

$$\Psi(f(\sigma)) \subseteq \{\psi'(\sigma') : \sigma' \in ES \cup PS\}.$$

   Moreover, by the definition of f, $\psi'(\sigma) \in \Psi(f(\sigma))$, so

$$\psi'(\sigma) \in \{\psi'(\sigma') : \sigma' \in ES \cup PS\}.$$

   Choose σ' in $ES \cup PS$ such that $\psi'(\sigma) = \psi'(\sigma')$. Then $\psi'(\sigma') \in \Psi(f(\sigma))$ and, since distinct zones in d represent disjoint sets, we deduce that $f(\sigma') = f(\sigma)$, as required.

4. Lastly, we show that if two positive or negative spiders, σ_1 and σ_2, have the same label then they have the same zone selected. Trivially,

$$\psi'(\sigma_1) = \psi'(\sigma_2)$$

   in other words σ_1 and σ_2 represent the same element. Therefore, both σ_1 and σ_2 represent an element in the set denoted by some zone z. Thus $f(\sigma_1) = f(\sigma_2) = z$ as required.

Hence all four conditions are met by f and we deduce that f is a zone selection function. Therefore d is consistent. Thus, d is consistent if and only if d is satisfiable. □

To conclude this section, we add an inference rule for inconsistency:

Inference Rule 1 (Inconsistency)**.** *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an inconsistent enhanced spider diagram. Let d' be any enhanced spider diagram. Then d may be replaced by d'.*

5 Inference Rules for Spiders

The goal of this section is to introduce inference rules that can later be used to reduce clutter in enhanced spider diagrams. These inference rules focus on spiders only. It is therefore helpful to introduce transformations on diagrams that remove and add spiders. In what follows we use $|$ to indicate a domain restriction.

Transformation 1. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. Let σ be a spider in $S(d)$. We define a spider removal operation on d:*

$$d - \sigma = (L, Z, ShZ, ES\backslash\{\sigma\}, PS\backslash\{\sigma\}, NS\backslash\{\sigma\}, \eta', \rho')$$

where $\eta' = \eta|_{S\backslash\{\sigma\}}$, and $\rho' = \rho|_{(PS\cup NS)\backslash\{\sigma\}}$.

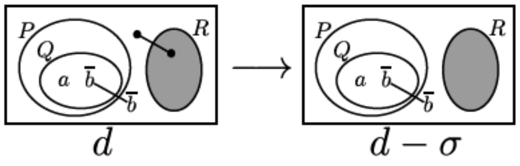


Figure 5. Removing a spider.

For example, in figure 5, the existential spider, σ, is removed from d to give $d - \sigma$. This removal transformation only needs to 'know' which spider to remove. However, when adding a spider, we need to know the location in which it is to be placed and, if it is positive or negative, its label must be supplied. Figure 6 shows three applications of the spider addition transformation. In each case, a spider is added to the region $r = \{(\{P\}, \{Q, R\}), (\emptyset, \{P, Q, R\})\}$. In the first case, the existential spider σ_e is added. In the second and third cases, a positive and, respectively, negative spider (σ_a and $\sigma_{\overline{a}}$ resp.) is added with the label a.

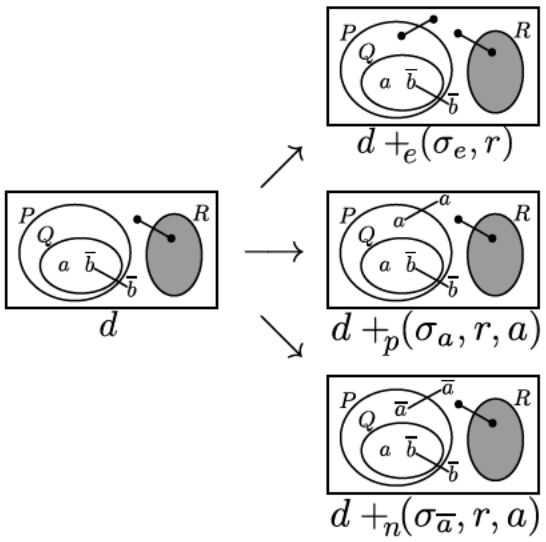


Figure 6. Adding spiders.

Transformation 2. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. Let σ be an element that is not in $S(d)$ (i.e. a fresh spider), let c be a constant in $\mathcal{C}$, and let r be a subset of Z. We define three spider addition operations on d:*

1. *$d +_e (\sigma, r) = (L, Z, ShZ, ES \cup \{\sigma\}, PS, NS, \eta \cup \{(\sigma, r)\}, \rho)$*
2. *$d +_p (\sigma, r, c) = (L, Z, ShZ, ES, PS \cup \{\sigma\}, NS, \eta \cup \{(\sigma, r)\}, \rho \cup \{(\sigma, c)\})$*
3. *$d +_n (\sigma, r, c) = (L, Z, ShZ, ES, PS, NS \cup \{\sigma\}, \eta \cup \{(\sigma, r)\}, \rho \cup \{(\sigma, c)\})$.*

These transformations will now be used to define inference rules that delete spiders, shrink spiders, and swap spiders; we do not provide rules for adding spiders since such a transformation increases visual clutter and our focus is on reducing clutter. Importantly, all these rules are equivalences: the diagram to which the rule is applied has the same models as the resulting diagram. The fact that the rules are equivalences means we can explore different representations of information using different types of spider. In addition, the rules are only defined for consistent diagrams; when applied to inconsistent diagrams they need not be equivalences. In section 6 we will discuss the impact of negative spiders on clutter reduction.

Firstly we introduce three inference rules that allow spiders to be deleted. Clearly, deleting spiders allows clutter to be reduced. We start by observing that, in any diagram d, and for constant, c, in $\mathcal{C}$, the individual represented by c lies in the set represented by $Z\backslash EZ$. Moreover, the individual cannot lie in the set represented by $NZ(c, d)$, so we can make the stronger assertion that the individual lies in the set represented by $(Z\backslash EZ)\backslash NZ(c, d)$. We use this insight in the first rule, which focuses on existential spiders and exploits absence information.

For example, in figure 7, the existential spider, σ, in the zone $(\emptyset, \{P, Q\})$ can be deleted. It is the only spider in this non-shaded location and, moreover, the negative zones for c are precisely all of the zones except $(\emptyset, \{P, Q\})$. Therefore, on deleting σ the information that $(\emptyset, \{P, Q\})$ does not represent the empty set, which is directly asserted by σ, can be deduced from the three negative spiders labelled c. Neither of the other two existential spiders can be deleted. Deleting the existential spider in the shaded zone $(\{P\}, \{Q\})$ would not be sound: in models for d, this zone contains exactly two elements, deleting this spider results in a diagram in which all models require this zone to contain just one element. The other existential spider, which is placed in two zones, tells us that there is some element in the respective set that is different from the individual b. Thus, deleting this other existential spider, whilst sound, weakens information and does not result in a semantically equivalent diagram.

Inference Rule 2 (Delete Existential Spider). *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram. Let σ be an existential spider in d. If*

1. *there are no other existential or positive spiders in d, whose location overlaps with σ's, that is for all σ' in $(ES \cup PS)\backslash\{\sigma\}$, $\eta(\sigma) \cap \eta(\sigma') = \emptyset$,*

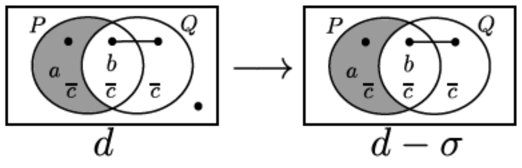


Figure 7. Delete existential spider.

2. *σ's location does not include shaded zones, that is*

$$\eta(\sigma) \cap ShZ = \emptyset,$$

and

3. *some constant, c, must represent an individual in the set denoted by $\eta(\sigma)$, that is*

$$(Z \backslash EZ) \backslash NZ(c,d) \subseteq \eta(\sigma).$$

then d may be replaced by $d-\sigma$.

Lemma 7. *Delete existential spider is sound and results in a semantically equivalent diagram.*

Proof. Suppose we delete existential spider σ from d to give $d-\sigma$. It is trivial to show that d semantically entails $d-\sigma$, in part since the location of σ does not include any shaded zones. Therefore, we focus on the converse: $d-\sigma$ semantically entails d. Let $I=(U,\psi,\Psi)$ be a model for $d-\sigma$ with valid $\psi'\colon (ES\backslash\{\sigma\}) \cup PS \cup NS \to U$. We show that I is a model for d. Our strategy is as follows:

1. show $\Psi(\eta(\sigma)) \neq \emptyset$,
2. choose $e \in \Psi(\eta(\sigma))$ and extend ψ' to ψ'' where $\psi''(\sigma)=e$, and
3. show ψ'' is valid for d.

Firstly, then, we show show $\Psi(\eta(\sigma)) \neq \emptyset$. Consider the constant, c, that ensures

$$(Z \backslash EZ) \backslash NZ(c,d) \subseteq \eta(\sigma).$$

We know that $\psi(c)$ is an element of U and, by lemma 4, $\psi(c) \in \Psi(z)$ for some $z \in Z$. Moreover, by lemma 2, z is not in $EZ(d)$. Thus far, we know that z is in $Z \backslash EZ$. By lemma 3 we further know that $\psi(c)$ is not in $\Psi(NZ(c,d))$. Since distinct zones represent disjoint sets, we deduce that z is not in $NZ(c,d)$. Hence,

$$z \in (Z \backslash EZ) \backslash NZ(c,d) \subseteq \eta(\sigma)$$

so

$$\psi(c) \in \Psi(z) \subseteq \Psi(\eta(\sigma)).$$

It follows that $\Psi(\eta(\sigma)) \neq \emptyset$.

Choose $e \in \Psi(\eta(\sigma))$ and define $\psi''\colon ES \cup PS \cup NS \to U$ by

$$\psi''(\sigma') = \begin{cases} \psi'(\sigma') & \text{if } \sigma' \neq \sigma \\ e & \text{otherwise.} \end{cases}$$

We show ψ'' is valid for d, but first we consider the missing zones condition. Trivially, this holds for d in I as it is identical for d and $d-\sigma$. Consider now the shaded zones condition. Let z be a shaded zone in d. Then z is shaded in $d-\sigma$. By the shaded zones condition for $d-\sigma$ we know

$$\Psi(z) \subseteq \{\psi'(\sigma') : \sigma' \in (ES \backslash \{\sigma\}) \cup PS\}.$$

Therefore

$$\Psi(z) \subseteq \{\psi''(\sigma') : \sigma' \in ES \cup PS\}$$

as required. Hence the shaded zones condition holds for d.

For the spider distinctness condition, let σ_1 and σ_2 be distinct existential or positive spiders in d. If both of them are in $d-\sigma$ then we know $\psi'(\sigma_1) \neq \psi'(\sigma_2)$ implying that $\psi''(\sigma_1) \neq \psi''(\sigma_2)$. Assume without loss of generality, that $\sigma_2 = \sigma$. We must show that $\psi''(\sigma_1) \neq \psi''(\sigma)$. We know already that $\psi''(\sigma)$ is in $\Psi(\eta(\sigma))$. Since ψ' is valid for $d-\sigma$, we also know that

$$\psi'(\sigma_1) = \psi''(\sigma_1) \in \Psi(\eta(\sigma_1)).$$

In d, by the definition of the delete existential spider inference rule, the spider σ has a location that is disjoint from all other spider locations. Therefore

$$\eta(\sigma_1) \cap \eta(\sigma) = \emptyset.$$

Thus

$$\psi''(\sigma_1) \notin \Psi(\eta(\sigma))$$

and we also have

$$\psi''(\sigma) = e \in \Psi(\eta(\sigma)).$$

It follows that $\psi''(\sigma_1) \neq \psi''(\sigma)$, as required. Hence the spider distinctness condition holds for d.

Trivially, the existential spiders condition holds for d, noting that it is identical to that for $d-\sigma$ except for the extra spider σ (and we know, by the construction of ψ'' that $\psi''(\sigma) \in \Psi(\eta(\sigma))$). Similarly, the positive and negative spiders conditions hold for d, as they are identical conditions in d and $d-\sigma$. Hence ψ'' is valid for d and it follows that I models d. Therefore, delete existential spider is sound and $d-\sigma$ is semantically equivalent to d.

□

Intuitively, the delete existential spider rule can be applied when we know the element represented by c is in the set represented by $\eta(\sigma)$. However, it is important that no other existential or positive spiders have a location that overlaps with σ, essentially because negative spiders provide no distinctness information. The next rule, which allows the deletion of a positive spider, is similar: a positive spider can be deleted when the information it provides is also given by negative spiders:

Inference Rule 3 (Delete Positive Spider). *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram. Let σ be a positive spider in d. If*

1. *there are no other existential or positive spiders in d, whose location overlaps with σ's, that is for all σ' in $(ES \cup PS)\backslash\{\sigma\}$, $\eta(\sigma) \cap \eta(\sigma') = \emptyset$,*
2. *the only shaded zones in σ's location are in $NZ(\rho(\sigma), d)$, that is*

$$\eta(\sigma) \cap ShZ \subseteq NZ(\rho(\sigma), d),$$

and

3. *the negative zones for $\rho(\sigma)$ indicate that σ must represent an individual in the set denoted by $\eta(\sigma)$:*

$$(Z\backslash EZ)\backslash NZ(\rho(\sigma), d) \subseteq \eta(\sigma).$$

then d may be replaced by $d - \sigma$.

Lemma 8. *Delete positive spider is sound and results in a semantically equivalent diagram.*

Proof Sketch. The proof is similar to that for the delete existential spider inference rule, so we just provide a sketch to illustrate key differences. Suppose we delete positive spider σ from d to give $d - \sigma$. It is trivial to show that d semantically entails $d - \sigma$, in part since the location of σ does not include any shaded zones other than those which are negative for $\rho(\sigma)$. Therefore, we focus on the converse: $d - \sigma$ semantically entails d. Let $I = (U, \psi, \Psi)$ be a model for $d - \sigma$ with valid ψ': $ES \cup (PS\backslash\{\sigma\}) \cup NS \rightarrow U$. We need to show that I is a model for d. The strategy is as follows:

1. show $\Psi(\eta(\sigma)) \neq \emptyset$,

2. show $\psi(\rho(\sigma)) \in \Psi(\eta(\sigma))$ and extend ψ' to ψ'' where $\psi''(\sigma) = \psi(\rho(\sigma))$, and

3. show ψ'' is valid for d.

Noting the similarity to lemma 7, the rest of the details are straightforward. □

Interestingly, negative spiders can always be deleted when they have multi-zone locations: if a negative spider, σ, has location $\{z_1, z_2\}$ for example, then this spider expresses that $\psi(\rho(\sigma)) \notin \Psi(z_1)$ or $\psi(\rho(\sigma)) \notin \Psi(z_2)$ which is trivially true in any interpretation. Also, just as positive spiders could be deleted when their informational content was represented by negative spiders, negative spiders can be deleted when their information is provided by positive spiders, by shading or, even, by other negative spiders.

To illustrate, in figure 8 the negative spider, σ, labelled b is deleted. This deletion does not weaken information since the positive spider labelled b provides the same absence information, albeit in a different form: from the positive spider, we can deduce that b is absent from the set represented by $(\{Q\}, \{P\})$. In fact, from d, any one of the negative spiders can be deleted. In the case of $\overline{a}$, the zone in which it is located is shaded and contains no part of any other spider: it is an empty zone. Therefore, the shading alone tells us that a does not lie in the set represented by $(\{P\}, \{Q\})$, so deleting $\overline{a}$ loses no information. In the remaining case, there are two $\overline{c}$s occupying the same zone and either one of them (but not both) can be deleted whilst preserving the informational content of d.

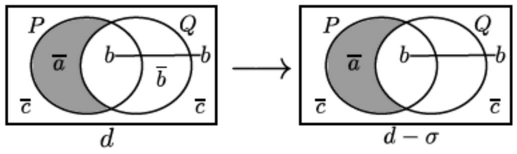


Figure 8. Delete negative spider.

Inference Rule 4 (Delete Negative Spider). *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram. Let σ be a negative spider in d. If*

1. *there is not a unique zone in $\eta(\sigma)$, that is*

$$|\eta(\sigma)| \neq 1$$

or

2. *there is a unique zone, z, in $\eta(\sigma)$, so $|\eta(\sigma)| = 1$, where either*
   (a) *the zone z is empty, that is $z \in EZ(d)$,*
   (b) *there exists a positive spider, σ', where $\rho(\sigma') = \rho(\sigma)$, whose location does not include z, that is $z \notin \eta(\sigma')$, or*
   (c) *there exists a negative spider, σ', where $\rho(\sigma') = \rho(\sigma)$, whose location is z, that is $\{z\} = \eta(\sigma')$*

then d may be replaced by $d - \sigma$.

Lemma 9. *Delete negative spider is sound and results in a semantically equivalent diagram.*

Proof. Firstly, it is trivial that deleting a negative spider is sound. Therefore we focus on shown that $d - \sigma$ semantically entails d. There are two cases, reflecting the definition of this inference rule: either (i) $\eta(\sigma)$ contains at least two zones or (ii) it contains a unique zone. In either case, we start assuming that $I = (U, \psi, \Psi)$ is a model for $d - \sigma$ with valid ψ': $ES \cup PS \cup (NS\backslash\{\sigma\}) \rightarrow U$. We need to show that

I is a model for d. We extend ψ' to ψ'': $ES \cup PS \cup NS \to U$ where ψ'' is identical to ψ' except that $\psi''(\sigma)$ is defined to be $\psi(\rho(\sigma))$. In either case (i) or (ii), it is trivial that all conditions for I to be a model hold for d except for the negative spiders condition.

Our task is to show that this condition holds. In the case (i) where $\eta(\sigma)$ contains at least two zones, it is trivial that there is a zone, z, in $\eta(\sigma)$ where $\psi''(\sigma) \notin \Psi(z)$ and we are done. In case (ii), $\eta(\sigma)$ contains exactly one zone. We show $\psi''(\sigma) \notin \eta(\sigma)$. There are three sub-cases, as in the definition of the delete negative spider inference rule.

(a) In this case, the zone, z, in $\eta(\sigma)$ is in $EZ(d)$. In this case, $\Psi(z) = \Psi(\eta(\sigma)) = \emptyset$, by lemma 2. Hence $\psi''(\sigma) \notin \Psi(\eta(\sigma))$ as required.

(b) Assume now that there is a positive spider, σ', where $\rho(\sigma') = \rho(\sigma)$. In which case, we know from the definition of the inference rule that $\eta(\sigma')$ does not include the unique zone, z, in $\eta(\sigma)$. From the positive spiders condition for $d - \sigma$, we know that $\psi'(\sigma') \in \Psi(\eta(\sigma'))$. Since distinct zones represent disjoint sets, we deduce that $\psi'(\sigma') \notin \Psi(\eta(\sigma))$. Since $\psi''(\sigma') = \psi'(\sigma') = \psi(\rho(\sigma'))$, $\rho(\sigma') = \rho(\sigma)$, and $\psi''(\sigma) = \psi(\rho(\sigma))$, it follows that $\psi''(\sigma) \notin \Psi(\eta(\sigma))$ as required.

(c) Finally, we consider the case where there exists a negative spider, σ', where $\rho(\sigma') = \rho(\sigma)$. In this case, we know that $\eta(\sigma')$ contains precisely the zone, z, in $\eta(\sigma)$. By lemma 3, it follows that

$$\psi(\rho(\sigma')) = \psi(\rho(\sigma)) \notin \Psi(z).$$

By definition we have $\psi''(\sigma) = \psi(\rho(\sigma))$, so

$$\psi''(\sigma) \notin \Psi(z)$$

as required.

Since ψ'' is otherwise identical to ψ' and the negative spiders condition holds for $d - \sigma$, the negative spiders condition holds for d. Hence I models d and it follows that the delete negative spiders inference rule is sound and produces a semantically equivalent diagram. □

Another way to reduce clutter arising from spiders is to remove zones from their locations. Focusing first on existential spiders, sometimes their locations can be shrunk when we have information provided by negative spiders. However, we can never remove shaded zones from their locations, as this would reduce the upper bound placed on the cardinality of the associated set and, thus, not be sound. Again, when defining this rule we must be mindful of the fact that negative spiders do not provide distinctness information: carelessly removing a zone from an existential spider's location could reduce the associated lower bound on set cardinality and would not result in an equivalent diagram.

Figure 9 illustrates how we can shrink an existential spider. Here, in d we can see that c must lie in the set represented by $(\{P, Q\}, \emptyset)$, due to the negative zones for c and the fact that c cannot be in $(\{P\}, \{Q\})$ due to the shading (this shaded zone is empty). Therefore, we can shrink the existential spider, removing the zone $(\{Q\}, \{P\})$ from its location without weakening information.

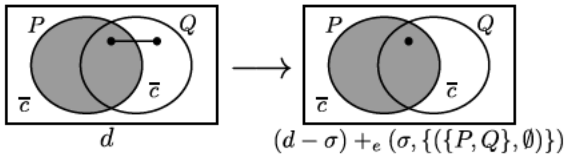


Figure 9. Shrink existential spider.

Inference Rule 5 (Shrink Existential Spider). *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram. Let σ be an existential spider in d occupying at least two zones of which one, z, is not shaded. If*

1. *there are no other existential or positive spiders in d, whose location overlaps with σ's, that is for all σ' in $(ES \cup PS) \backslash \{\sigma\}$, $\eta(\sigma) \cap \eta(\sigma') = \emptyset$, and*
2. *some constant, c, must represent an individual in the set denoted by $\eta(\sigma) \backslash \{z\}$, that is*

$$(Z \backslash EZ) \backslash NZ(c, d) \subseteq \eta(\sigma) \backslash \{z\}.$$

Then d may be replaced by $(d - \sigma) +_e (\sigma, \eta(\sigma) \backslash \{z\})$.

Lemma 10. *Shrink existential spider is sound and results in a semantically equivalent diagram.*

Proof. It is trivial to show that $d - \sigma +_e (\sigma, \eta(\sigma) \backslash \{z\})$ semantically entails d. We show that d semantically entails $d - \sigma +_e (\sigma, \eta(\sigma) \backslash \{z\})$. Let $I = (U, \psi, \Psi)$ be a model for d with valid ψ': $ES \cup PS \cup NS \to U$. We need to show that I is a model for $d - \sigma +_e (\sigma, \eta(\sigma) \backslash \{z\})$. It is trivial that the missing zones condition holds for $d - \sigma +_e (\sigma, \eta(\sigma) \backslash \{z\})$ since it holds for d. We focus on the remaining five conditions. There are two cases to consider, based on the zone, z', in which $\psi'(\sigma)$ represents an element. The first case is where $z' \neq z$ and the second case is where $z' = z$.

For the first case, we show that ψ' is valid for $d - \sigma +_e (\sigma, \eta(\sigma) \backslash \{z\})$. It is trivial that the shaded zones condition, spider distinctness condition, positive spiders condition and negative spiders condition hold for $d - \sigma +_e (\sigma, \eta(\sigma) \backslash \{z\})$ under ψ' since the conditions remain unchanged from those for d. What remains is to show that the existential spiders condition holds and here we only need to worry about σ since all other spider locations remain unchanged. By the existential spiders condition for d, we also know that

$$\psi'(\sigma) \in \Psi(\eta(\sigma))$$

so, since $z' \neq z$,

$$\psi'(\sigma) \in \Psi(\eta(\sigma)) \backslash \Psi(z).$$

Therefore

$$\psi'(\sigma) \in \Psi(\eta(\sigma) \backslash \{z\})$$

as required. Therefore when $z' \neq z$, I is a model for $d - \sigma +_e (\sigma, \eta(\sigma)\backslash\{z\})$.

We must now consider the case where $z' = z$. Here we must define an alternative mapping of spiders to elements, since ψ' will not ensure $\psi'(\sigma)$ represents an element in $\Psi(\eta(\sigma)\backslash\{z\})$. Noting, by the definition of the shrink existential spider rule, that c is a constant where

$$(Z\backslash EZ)\backslash NZ(c,d) \subseteq \eta(\sigma)\backslash\{z\}$$

we define ψ'': $ES \cup PS \cup NS \to U$ by

$$\psi''(\sigma') = \begin{cases} \psi'(\sigma') & \text{if } \sigma' \neq \sigma \\ \psi(c) & \text{otherwise.} \end{cases}$$

We now show that I is a model for $d - \sigma +_e (\sigma, \eta(\sigma)\backslash\{z\})$ using ψ''.

For the shading condition, we need to verify, for each shaded zone, z'', in ShZ that

$$\Psi(z'') \subseteq \{\psi''(\sigma') : \sigma' \in ES \cup PS\}.$$

It can readily be shown that

$$\Psi(z'') \subseteq \{\psi'(\sigma') : \sigma' \in ES \cup PS\} \backslash \{\psi'(\sigma)\}$$

because $\psi'(\sigma) \in \Psi(z)$, $z \neq z''$ and so $\psi'(\sigma) \notin \Psi(z'')$. Therefore

$$\Psi(z'') \subseteq \{\psi''(\sigma') : \sigma' \in ES \cup PS\} \backslash \{\psi'(\sigma)\}$$

so

$$\Psi(z'') \subseteq \{\psi''(\sigma') : \sigma' \in (ES\backslash\{\sigma\}) \cup PS\}.$$

Therefore

$$\begin{aligned} \Psi(z'') &\subseteq \{\psi''(\sigma') : \sigma' \in (ES\backslash\{\sigma\}) \cup PS\} \cup \{\psi''(\sigma)\} \\ &= \{\psi''(\sigma') : \sigma' \in ES \cup PS\} \end{aligned}$$

as required. Therefore the shaded zones condition holds.

For the spider distinctness condition, let σ_1 and σ_2 be spiders in $ES \cup PS$. If neither σ_1 nor σ_2 are the spider σ then it follows, by the spider distinctness condition for d, that

$$\psi''(\sigma_1) = \psi''(\sigma_2) \Rightarrow \sigma_1 = \sigma_2.$$

Suppose, without loss of generality, that $\sigma_1 = \sigma$. We must show that if $\psi''(\sigma) = \psi''(\sigma_2)$ then $\sigma = \sigma_2$. Assuming $\psi''(\sigma) = \psi''(\sigma_2)$, we have $\psi''(\sigma) = \psi(c) = \psi''(\sigma_2) = \psi'(\sigma_2)$. Consider $\psi(c)$. Since the missing zones condition holds, by lemma 4, $\psi(c) \in \Psi(Z)$. By lemma 2, it follows that

$$\psi(c) \in \Psi(Z\backslash EZ).$$

By lemma 3, we further deduce that

$$\psi(c) \in \Psi(Z\backslash EZ)\backslash\Psi(NZ(c,d)) = \Psi((Z\backslash EZ)\backslash NZ(c,d)).$$

By the definition of the shrink existential spider rule,

$$NZ(c,d) = (Z\backslash EZ)\backslash NZ(c,d) \subseteq \eta(\sigma)\backslash\{z\},$$

so

$$\psi(c) \in \Psi(\eta(\sigma)\backslash\{z\}) \subseteq \Psi(\eta(\sigma)).$$

We also have, by the existential and positive spiders condition for d

$$\psi(c) = \psi''(\sigma_2) = \psi'(\sigma_2) \in \Psi(\eta(\sigma_2)).$$

Therefore, since distinct zones represent disjoint sets,

$$\eta(\sigma) \cap \eta(\sigma_2) \neq \emptyset.$$

By the definition of the shrink existential spider rule, no positive or existential spider has a location that overlaps with $\eta(\sigma)$, other than σ itself. Therefore, $\sigma = \sigma_2$ as required. Hence the spiders distinctness condition holds for $d - \sigma +_e (\sigma, \eta(\sigma)\backslash\{z\})$.

Focusing now on the existential spiders condition for $d - \sigma +_e (\sigma, \eta(\sigma)\backslash\{z\})$, the only way this can fail is if $\psi''(\sigma) = \psi(c)$ is not in $\Psi(\eta(\sigma))$. We have just seen that

$$\psi(c) \in \Psi(\eta(\sigma)\backslash\{z\}) \subseteq \Psi(\eta(\sigma))$$

so it follows that the existential spiders condition holds for $d - \sigma +_e (\sigma, \eta(\sigma)\backslash\{z\})$. It is trivial that the positive and negative spiders conditions hold, since they are identical for d and $d - \sigma +_e (\sigma, \eta(\sigma)\backslash\{z\})$. Hence I is a model for d. It follows that the shrink existential spiders rule is sound and produces a semantically equivalent diagram. □

In the above rule, we know that the set represented by $\eta(\sigma)\backslash\{z\}$ must contain $\psi(c)$ so it is not empty. It is therefore possible to shrink σ, removing z, without weakening information, in part since z is not shaded and in part since no other existential or positive spider has a location that overlaps with σ. We can also shrink positive spiders, when their locations include a negative zone.

Inference Rule 6 (Shrink Positive Spider)**.** *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram. Let σ be a positive spider in d occupying at least two zones where $\eta(\sigma) \cap NZ(\rho(\sigma), d) \neq \emptyset$. Let $z \in \eta(\sigma) \cap NZ(\rho(\sigma), d)$. Then d may be replaced by $(d - \sigma) +_p (\sigma, \eta(\sigma)\backslash\{z\}, \rho(\sigma))$ and vice versa.*

Lemma 11. *Shrink positive spider is sound and results in a semantically equivalent diagram.*

Proof Sketch. It is trivial to show that $d - \sigma +_p (\sigma, \eta(\sigma)\backslash\{z\}, \rho(\sigma))$ semantically entails d. We show that d semantically entails $d - \sigma +_p (\sigma, \eta(\sigma)\backslash\{z\}, \rho(\sigma))$. Let $I = (U, \psi, \Psi)$ be a model for d with valid $\psi'\colon ES \cup PS \cup NS \to U$. We need to show that I is a model for $d - \sigma +_p (\sigma, \eta(\sigma)\backslash\{z\}, \rho(\sigma))$. In this case, it is straightforward to show that ψ' is valid for $d - \sigma +_p (\sigma, \eta(\sigma)\backslash\{z\}, \rho(\sigma))$, using lemma 3 to verify that the positive spiders condition holds. □

It is also possible to define a rule for shrinking negative spiders (when they include at least three zones in their locations). However, we have already seen that negative spiders with multiple zone locations can be deleted without losing information. Therefore we do not need a shrink negative spider inference rule in order to explore clutter reduction.

Lastly, we consider when it is possible to swap between different types of spider. Sometimes it is sound to swap an existential spider for a positive spider with the same location but this has no material impact on clutter so we omit this case. It is not sound to swap an existential spider, σ, for negative spiders as this would either reduce the lower bound on the set denoted by the location of σ or introduce new information about some specific individual. Therefore, there is only one interesting case where we can swap between types of spider: swapping between positive and negative spiders can alter the visual clutter in enhanced spider diagrams.

Swapping spiders is illustrated in figure 10, where the positive spider, σ, namely $a - a - a$, is swapped for two negative spiders, σ_1 and σ_2. These negative spiders occupy the two non-empty (single zone) regions $r_1 = \{(\{Q, R\}, \{P\})\}$ and $r_2 = \{(\{Q\}, \{P, R\})\}$ that did not previously contain negative a spiders. It is clear to see the information that a is in the set $U\backslash(Q \cup R)$ is not lost when this swap is performed.

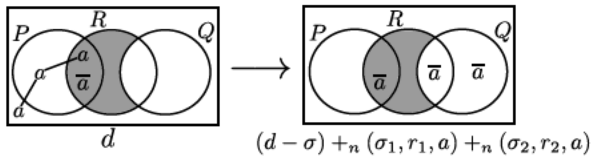


Figure 10. Swapping spiders.

Inference Rule 7 (Swap Positive and Negative Spiders). *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be a consistent enhanced spider diagram. Let σ be a positive spider in d where*

1. *there are no other existential or positive spiders in d, whose location overlaps with σ's, that is for all σ' in $(ES \cup PS)\backslash\{\sigma\}$, $\eta(\sigma) \cap \eta(\sigma') = \emptyset$, and*
2. *the only shaded zones in σ's location are in $NZ(\rho(\sigma), d)$, that is*

$$\eta(\sigma) \cap ShZ \subseteq NZ(\rho(\sigma), d).$$

Let $\sigma_1, ..., \sigma_m$ be m fresh spiders, one for each zone in

$$Z\backslash(\eta(\sigma) \cup NZ(\rho(\sigma), d) \cup EZ) = \{z_1, ..., z_m\}.$$

Then d may be replaced by

$$(d - \sigma) +_n (\sigma_1, \{z_1\}, \rho(\sigma)) +_n .... +_n (\sigma_m, \{z_m\}, \rho(\sigma))$$

and vice versa.

Lemma 12. *Swapping positive and negative spiders is sound and results in a semantically equivalent diagram.*

Proof Sketch. Firstly, it is straightforward to show that

$$d +_n (\sigma_1, \{z_1\}, \rho(\sigma)) +_n .... +_n (\sigma_m, \{z_m\}, \rho(\sigma))$$

is semantically equivalent to d; intuitively, negative spiders are added to non-empty zones that are not in the location of σ. The spider σ can be deleted from

$$d +_n (\sigma_1, \{z_1\}, \rho(\sigma)) +_n .... +_n (\sigma_m, \{z_m\}, \rho(\sigma),$$

using the delete positive spider rule to give, by lemma 8, the semantically equivalent diagram,

$$d +_n (\sigma_1, \{z_1\}, \rho(\sigma)) +_n .... +_n (\sigma_m, \{z_m\}, \rho(\sigma) - \sigma,$$

. Noting the commutativity of the transformations, this diagram is

$$(d - \sigma) +_n (\sigma_1, \{z_1\}, \rho(\sigma)) +_n .... +_n (\sigma_m, \{z_m\}, \rho(\sigma)).$$

Therefore, swapping positive and negative spiders is sound and results in a semantically equivalent diagram. □

The lemmas in this section combine to establish that all of the inference rules are sound:

Theorem 3. *The inference rules are all sound and result in semantically equivalent diagrams.*

6. Measuring and Reducing Clutter

The clutter measure given in [3] readily generalizes to enhanced spider diagrams. At the drawn diagram level, the measure counts the number of nodes and the lines used to connect spider nodes[2]. For each spider, the number of lines

[2]Recall that this measure of clutter was empirically evaluated in [16] where it was found that high levels of clutter resulted in worse task performance.

is one less than the number of zones in its location. For example, in figure 2, d_3 has a clutter score of 13, since there are seven nodes and six connecting lines, whereas d_4 has a score of 1. In figure 4, d_7 has a score of 0 but the semantically equivalent diagram, d_8, has a score of 11.

Definition 11. *Let $d = (L, Z, ShZ, ES, PS, NS, \eta, \rho)$ be an enhanced spider diagram. The* ***spider clutter score*** *for d, denoted $\mathcal{SCS}(d)$, is*

$$\mathcal{SCS}(d) = \sum_{\sigma \in S(d)} (2|\eta(\sigma)| - 1).$$

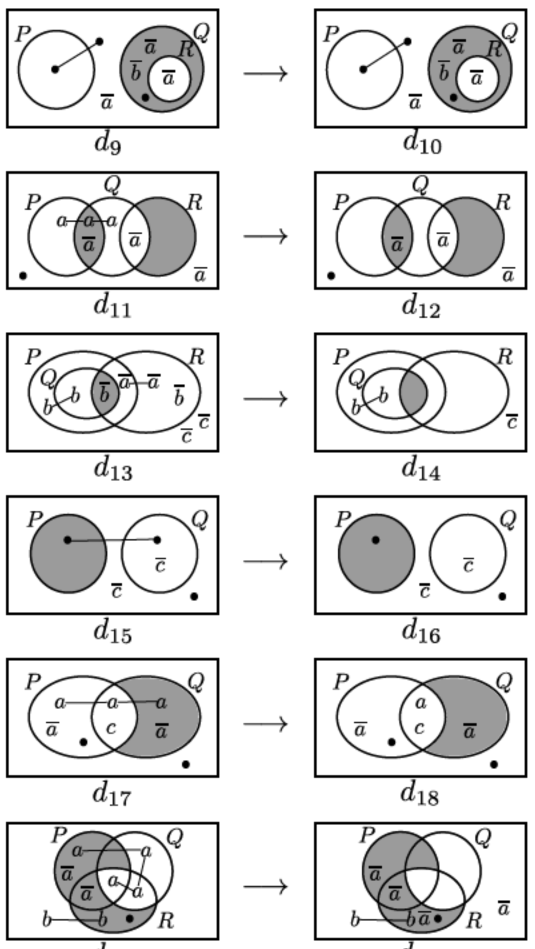


Figure 11. Rules' impact the clutter score.

We now demonstrate how the rules can impact the clutter score. In figure 11, the first three rows illustrate the three deletion rules. Deleting the existential spider located in two zones in d_9 to give d_{10} reduces the score by 3. This spider can only be deleted, without losing information, since the negative spiders express that a is absent from Q. The region outside Q therefore 'contains' a and it is precisely this region that contains the existential spider that is deleted to give d_{10}. Deleting the positive spider, a, from d_{11} to give d_{12} reduces the clutter score by 5. This time, the information provided about the individual a by the positive spider can be inferred from the negative spiders, $\overline{a}$, and so a is redundant. These two rules clearly show that absence information, provided by negative spiders, leads to a reduction in visual clutter. When it comes to deleting negative spiders, there are various cases illustrated in d_{13} and d_{14}. Deleting $\overline{a} - \overline{a}$ reduces the score by 3, deleting two $\overline{b}$s by 2 and one $\overline{c}$ by 1 (total: 6).

The next two rows show applications of shrinking rules. Without absence information, it is not possible to shrink spiders and maintain the semantic information. In d_{15}, we can infer that $c \in P \backslash Q$, since $c \notin Q \backslash P$ and $c \notin \overline{P \cup Q}$. Thus, the existential spider in d_{15} must represent the same element as c, and be in P. Therefore we can remove a zone from its location, as shown in d_{16}, reducing the score by 2. The case for d_{17} is more straightforward: the absence information about a allows us to reduce the location of the $a - a - a$ spider, lowering the clutter score by 4 after two applications of the shrink positive spider rule.

Lemma 13. *A single application of any one of the three deletion rules reduces the clutter score by $2|\eta(\sigma)| - 1$ where σ is the deleted spider.*

Lemma 14. *A single application of any one of the two shrinking rules reduces the clutter score by 2 where σ is the shrunk spider.*

The last row of figure 11 shows an application of the swap rule. The $a - a - a - a$ spider is swapped for two $\overline{a}$ spiders, reducing the clutter score by 5.

Lemma 15. *A single application of the swap rule reduces clutter whenever*

$$2|\eta(\sigma)| - 1 > |Z \backslash (\eta(\sigma) \cup NZ(\rho(\sigma), d) \cup EZ)|,$$

where σ is the positive spider to be swapped.

7 Discussion on Clutter Reduction

We now discuss how clutter reduction using absence, conveyed via negative spiders, contrasts with the Venn-i^e case. In enhanced spider diagrams, many of the rules that we have introduced required the subject existential or positive spider, σ, to be the only one in its location. This is because to delete, shrink or swap such a spider, absence information must be used, yet negative spiders do not provide distinctness information. Considering an absence spider in isolation, it only indicates the set in which the represented individual does not lie and, therefore, in which set it does lie. Of course, this is identical for $\overline{i}$-sequences in

Venn-i^e. However, a significant difference is that, in Venn-i^e, $\otimes$-sequences and i-sequences do not represent distinct elements, so Venn-i^e's inference rules are much less restrictive.

An example can be seen in figure 12. Firstly, we note that D uses an $\otimes$-sequence, which asserts that the set $R\backslash P$ is not empty, as well as i-sequences (for presence) and $\overline{i}$-sequences (for absence). To create D_1, the $\overline{a}$-sequences are swapped for an a-sequence, a transformation which in general can increase clutter. This results in a diagram where all information about the sets in which particular individuals lie is given using presence information. The diagram D_2 is created by iteratively shrinking a-sequences, as we can deduce from D_1 that a is in the set $R \cap \overline{P} \cap \overline{Q}$. The next step shrinks the $\otimes$-sequence: in Venn-i^e's, shaded regions always represent the empty set, so we can remove the $\otimes$ symbol from the shaded zone, shrinking the sequence. Next, we notice that, since a represents an individual in the set $R \cap \overline{P} \cap \overline{Q}$, the $\otimes$-sequence is redundant (here, it is important to recall that distinct sequences do not denote distinct individuals in Venn-i^e. Lastly, we can swap $b - -b - -b$ for a single $\overline{b}$, giving us D_{min}. The diagram D_{min} is semantically equivalent to D and is minimally cluttered. It should be evident from this example that not asserting the distinctness of individuals has led to more freedom in when inference rules can be applied than we saw for enhanced spider diagrams, albeit at the expense of expressive power.

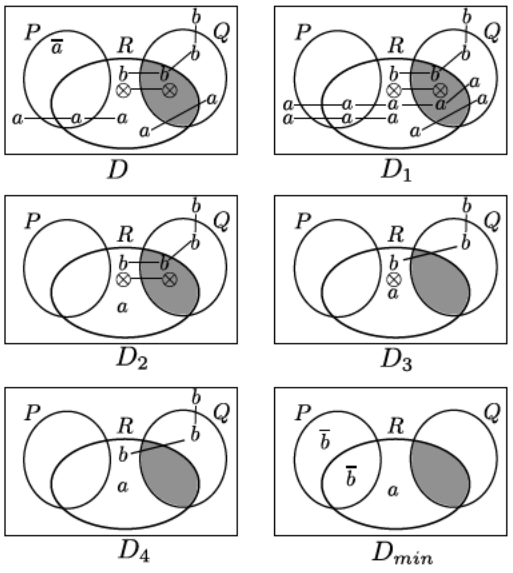


Figure 12. Rules' impact the clutter score.

In summary, as a consequence of the key differences between Venn-i^e and enhanced spider diagrams, there are many more situations in Venn-i^e in which information about the absence of individuals can be used to reduce clutter. A less diagrammatic interpretation of absence, whereby negative spiders *do* assert that the represented individual is distinct from those represented by the other spiders, would lead to the ability to reduce clutter further. This will be an interesting avenue for future work, particularly with regard to the usability of the resulting logic relative to enhanced spider diagrams.

8. Conclusion

We have introduced enhanced spider diagrams, which include syntax for visually representing the absence of individuals from particular sets directly. Necessary and sufficient conditions for diagram satisfiability were given, since unsatisfiable diagrams are semantically equivalent to diagrams with a zero clutter score. Following from this, we defined inference rules that permitted clutter to be reduced in satisfiable diagrams. Interestingly, the level of clutter reduction that is possible in this system is not as dramatic as was seen for the Venn-i^e system, where i-sequences and $\overline{i}$-sequences could readily be swapped to alter clutter. As indicated, an alternative, less well-matched, interpretation of negative spiders could lead to larger reductions in visual clutter. However, it is unclear whether lower clutter, with a less well-matched syntax, or higher clutter with a well-matched syntax is most effective for human cognition. Such a trade-off should be explored as it could provide important insight into choices that must be made when designing diagrammatic systems. Future work should also seek to understand the impact of clutter reduction on user task performance. Both of these aspects are important for ensuring an effective system of logic, accessible to users of the notation, is produced.

Acknowledgement This research was conducted whilst Gem Stapleton was Visiting Professor at Jadavpur University and is partially funded by a Leverhulme Trust Research Project Grant (RPG-2016-082) for the project entitled Accessible Reasoning with Diagrams.

References

[1] M. Alqadah, G. Stapleton, J. Howse, and P. Chapman. Evaluating the impact of clutter in Euler diagrams. In *8th Int. Conf. on Diagrams*, pages 109–123. Springer, 2014.

[2] R. Bhattacharjee, M. Chakraborty, and L. Choudhury. Venn diagram with names of individuals and their absence: A non-classical diagram logic. *Logica Universalis*, pages 1–66, 2018.

[3] J. Burton, M. Chakraborty, L. Choudhury, and G. Stapleton. Minimizing clutter using absence in Venn-i^e. In *9th Int. Conf. on Diagrams*, pages 107-122. Springer, 2016.

[4] L. Choudhury and M. K. Chakraborty. On extending Venn diagrams by augmenting names of individuals. In *3rd Int. Conf. on Diagrams*, pages 142–146. Springer, 2004.

[5] J. Gil, J. Howse, and S. Kent. Formalising spider diagrams. In *IEEE Symposium on Visual Languages (VL99), Tokyo*, pages 130–137. 1999.

[6] C. Gurr. Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. *J. Visual Languages and Computing*, 10(4):317–342, 1999.

[7] L. Horn. *A Natural History of Negation*. CSLI Lecture Notes. Center for the Study of Language and Information, 2001.

[8] T. Hou, P. Chapman, and A. Blake. Antipattern comprehension: An empirical evaluation. In *9th Int. Conf. on Formal Ontology in Information Systems*, pages 211–224, 2016.

[9] T. Hou, P. Chapman, and I. Oliver. Measuring perceived clutter in concept diagrams. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 31–39. 2016.

[10] J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS J. Computation and Mathematics*, 8:145–194, 2005.

[11] J. Howse, G. Stapleton, K. Taylor, and P. Chapman. Visualizing ontologies: A case study. In *International Semantic Web Conference*, pages 257–272. Springer, 2011.

[12] C. John, A. Fish, J. Howse, and J. Taylor. Exploring the notion of clutter in Euler diagrams. In *4th Int. Conf. on Diagrams*, pages 267–282, 2006. Springer.

[13] C. Peirce. *Collected Papers*, volume 4. Harvard University Press, 1933.

[14] Y. Sato, K. Mineshima, and R. Takemura. The Efficacy of Euler and Venn Diagrams in Deductive Reasoning: Empirical Findings. In *6th Int. Conf. on Diagrams*, pages 6–22. Springer, 2010.

[15] S.-J. Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994.

[16] G. Stapleton, A. Blake, J. Burton, and A. Touloumis. Presence and absence of individuals in diagrammatic logics: An empirical comparison. *Studia Logica*, 105(4):787–815, 2017.

[17] G. Stapleton, L. Choudhury, and M. Chakraborty. Spider diagrams with absence. In *DMS VIVA 2018*. KSI, 2018.

[18] G. Stapleton, J. Taylor, J. Howse, and S. Thompson. The expressiveness of spider diagrams augmented with constants. *J. Visual Languages and Computing*, 20:30–49, 2009.

[19] N. Swoboda and G. Allwein. Using DAG transformations to verify Euler/Venn homogeneous and Euler/Venn FOL heterogeneous rules of inference. *J. Software and System Modeling*, 3(2):136–149, 2004.

Dominant Colors as Image Content Descriptors: A Study with Users

Soraia M. Alarcão Ruben Pavão Manuel J. Fonseca
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
salarcao@lasige.di.fc.ul.pt, rubenpavao@gmail.com, mjfonseca@ciencias.ulisboa.pt

***Abstract*—Image content are typically described using low level features such as color, texture, shape, or a combination of the previous. A particular use of color is the identification of the dominant colors in images to describe its content, for image retrieval, for instance. In this paper, we present a study with users to verify if the dominant colors can be used as image content descriptors. From the study we identified the dominant and the search colors users associated to a set of images. We supplemented this information with gaze coordinates, collected with an affordable eye tracker, to register the regions at which people looked while identifying colors in the images. The analysis of the data revealed that users used a small set of color names, and that the colors used for searching were similar to those considered dominant, validating the use of dominant colors as image descriptors. As a result of the study, we make available a dataset of 100 images annotated with their dominant colors, the colors that users would use to search for them, and the areas where they looked while identifying both types of colors.**

I. Introduction

Color is one of the most distinctive visual features. Various systems for exploring, searching and presenting images to users take advantage of it through the use of image's dominant colors. Although there are mechanisms to search for or explore images through their dominant colors, these are usually identified from the perspective of the system and making several assumptions (e.g. more importance to the center, salient objects, etc.) and not based on the human perception of colors. That is, typically the dominant color is the one that occupies the largest area of the image. However, from the point of view of people, the dominant colors are not always those that cover more pixels. Additionally, most works consider too many colors as possible dominant colors, making their naming almost impossible, when users may want/need to explore or retrieve images by specifying the colors names.

The aim of this paper is to investigate whether dominant colors can be used as image content descriptors, and whether there is a relation between the regions at which people look and the colors they identify. To that end, we conducted a study with 40 participants in our research lab.

In particular, we designed two setups, one where we asked participants to identify up to three dominant colors in an image, and another where we asked them to mention up to three colors they will use to search for the presented image. Additionally, we collected eye tracking data of the regions of the image at which users looked while identifying colors.

DOI Reference Number: 10.18293/VLSS2018-034

From the data collected, we found that the colors used for searching are similar to those considered dominant, which means that we can develop a retrieval system where images are described by their dominant colors. In terms of the gaze information, we can conclude that there is no strong correlation between the regions at which people looked and the colors they identified. In most cases users looked at one region (e.g. faces) and mentioned a color that is presented in another region (e.g. t-shirt).

Our contributions are: 1) the confirmation of the JNS 11 colors as a valid reduced set of colors; 2) a dataset of 100 images annotated with their dominant colors and search colors identified by people; 3) gaze coordinates of users while identifying dominant and search colors in images.

II. Background and Related Work

In this section, we provide some background about color perception, color naming, dominant colors, and the use of eye-tracking to identify where people look at in images.

A. Color Perception

Color is the perceptual phenomenon related to the spectral characteristics of the electromagnetic radiation in the visible wavelengths (approximately from 380-750 nm). As suggested by human visual perception research [1] color is considered a pre-attentive property known to attract our visual attention above and beyond other object properties such as shape.

Our vision starts on the eye retina with two types of photoreceptors that receive the light stimulus and emit electrical impulses. Rods are responsible to operate at low light levels (scotopic vision), while cones operate at higher light levels (photopic vision). Cones are the ones responsible for the color vision, having a high spatial acuity. These electrical signals are then processed in the cortex, with our previously accumulated visual experience (memory), to form representations (Visual Perception) of color, shape, movement etc.

As so, we can say that color is the result of interpretation in the brain of the perception of light in the human eye and our visual memory.

B. Color Naming

In everyday life, we mainly identify colors by their names, which requires a general color vocabulary that is far from being precise. Given the importance of color naming, a variety of models and studies describing how people associate names and colors were introduced. Berlin and Kay studied the color naming behavior with subjects from multiple languages [2]. They concluded that the basic color terms in a culture can be predicted by the number of color terms the culture has. For English, they identified the following 11 basic terms: black, white, red, green, yellow, blue, brown, pink, orange, purple, and gray. Mojsilovic *et al.* presented a computational model for color categorization and naming of the 11 basic colors plus beige and olive [3].

Weijer *et al.* used real-world images to learn the 11 basic colors [4]. Moroney *et al.* conducted an unconstrained web-based study where they identified the 20 most commonly used color terms: green, blue, purple, red, pink, light, lime, dark blue, brown, yellow, black, orange, sky, bright, violet, olive, navy, sea, teal, and royal [5]. Menegaz *et al.* proposed a discrete model for color naming, where each of the 11 basic color terms was modeled as a fuzzy set [6]. Benavente *et al.* presented a parametric model for automatic color naming, where each of the 11 basic color terms was modeled as a fuzzy set with a parametric membership function [7].

As we can see, various authors adopted the set of 11 colors proposed by Berlin and Kay, probably because it is considered to contain colors that can be named by all cultures. Indeed, in 2000, Chang *et al.* coined it as the "Just Not the Same" colors (JNS), because any two colors from this set are not perceived as the same [8].

C. Dominant Colors

In general, color is a very distinctive feature, and as such several image search systems take advantage of it. In particular, they use the dominant colors of the images as a mechanism to describe and index their content. Usually, these systems rely mostly on color histograms to provide both the description of the colors present in an image and their quantities. Histograms are obtained by counting the number of pixels for each color, after quantizing the image colors into a reduced set of colors.

The VisualSEEk was one of the first systems for searching images using the dominant colors. It used the HSV color space to compute a histogram of 166 colors, from which it identified the dominant ones [9]. Deng *et al.* presented a feature descriptor that uses segmentation and color clustering to identify representative colors in each image's region [10]. Mojsilovic *et al.* proposed a method to compute dominant colors by considering both information captured through the image histogram and extracted from spatial relationships between frequently occurring colors [11].

Atsalakis *et al.* proposed the use of a neural network to automatically identify the significant colors with the minimum number of color classes [12]. Younnes *et al.* [13] and Amante *et al.* [14] proposed methods based on a fuzzy representation of colors to identify the dominant colors. Talib *et al.* proposed a method to reduce the background effect on the computation of dominant colors. Authors assigned weights to each dominant color in accordance with its belonging to the object or the background. The background colors, which are in contact with the image borders and out of salient object area, received a lower weight [15].

Although there are mechanisms for content-based image retrieval using dominant colors, most of them identify the dominant colors from the perspective of the system and not taking into consideration the human perception of colors.

D. Eye-tracking

Eye-tracking consists on cameras continuously tracking the position or orientation of the eyes [16]. Fixation consists on maintaining the visual gaze on a single location, and is useful to determine the focus of attention, i.e., to identify what triggered the attention change. Datasets of images annotated with eye-tracking information are important for the development of saliency models, i.e., to identify which information on an image attracts visual attention from the person looking at it.

In Table I, adapted from [17], we present some of the existing datasets available in the public domain (for detailed information, see [18], [17]). As far as we know, all of them contain eye tracking information but none is related to the tasks of looking at images while identifying the dominant colors or the colors to be used for searching.

Table I
DATASETS OF IMAGES ANNOTATED WITH EYE-TRACKING INFORMATION.

	Fixations		Inter-Fixation	Raw
	Locations	Durations	Durations	Data
DUT-OMRON	yes			
GazeCom Image				yes
MIT CSAIL				yes
MIT LowRes				yes
VAIQ				yes
IRCCyN Image 1	yes	yes		
Memorability	yes	yes		
McGill ImgSal	yes			yes
KTH	yes	yes	yes	
FiFA	yes	yes		yes
LIVE DOVES	yes	yes		yes
MIT CVCL	yes	yes		yes

III. User Study

In this section, we describe the study carried out to collect information about the way users identify colors in images (both for searching and as dominant), the names of colors they mention, and for what regions of the image they look while enumerating the colors.

A. Participants

Forty participants, divided into two groups of 20, completed the study. The first group (G1) was composed of 14 males and 6 females, with an average of 22 years old (SD=2.86). Six users wore glasses and one wore contact lenses. In the second group (G2) there were 12 males and 8 females, with an average of 21 years old (SD=2.96). Seven wore glasses and two contact lenses. All participants were voluntaries and had never used an eye tracker. Participants from group G1 answered question Q1 *"What are the (up to) three colors that you identify as dominant in this image?"*, while participants from group G2 responded to question Q2 *"What (up to three) colors would you use to search for this image?"*.

B. Apparatus and Material

We used a desktop computer with an application to present the images to the users and register the gaze coordinates collected by the eye tracker. We used TheEyeTribe (an affordable eye tracker), placed under a 20" LCD monitor with a resolution of 1600 x 900 pixels. To collect the coordinates, we used the eye tracker API with the maximum sampling rate supported (60 Hz). Participants were placed at a distance between 50 cm to 70 cm of the monitor (and the eye tracker). All users used the same computer and eye tracker, in the same place, with the same setup.

For the study, we used a set of 100 images (all with Creative Commons licensing) collected from Flickr, and organized into 30 categories: animal, architecture, baby, beach, bird, building, car, clouds, dog, flowers, food, girl, graffiti, lake, landscape, nature, night, people, portrait, river, sea, sign, sky, snow, street, sun, sunset, trees, urban, and water. These categories were based on the ones used in the MIRFLICKR dataset. We did not use this dataset because its images have a reduced size (500 x 500 pixels), which would produce poor results for the gaze coordinates.

To gather the images for our dataset, we performed an advanced search on Flickr, using the category name as tag and "Large" as the minimum size. For each category we selected four images (the first, third, fifth and seventh). After this initial step, we ended up with 120 images. From these, we discarded 20 images that were very similar to others in the dataset, thus getting 100 images. All images were resized, keeping the aspect ratio, to have their width or height equal to the width (1600) or height (900) of the screen (e.g. 1350 x 900; 669 x 900). By doing this, we had a direct correspondence between the images and the screen (and eye tracker) coordinates.

C. Research Questions

Taking into consideration the goals of our study, we identified six research questions that we wanted to answer:

- RQ_1 Can we reduce the name of all mentioned colors to a small subset (palette) of colors?
- RQ_2 Do users use the colors they consider dominant in an image to search for it?
- RQ_3 Where do people look at more often in an image while mentioning its colors?
- RQ_4 Do users look at the regions where the mentioned colors are?
- RQ_5 Does the category of the image affect the gaze pattern of the users?
- RQ_6 Does the type of color (e.g. warm, pure, etc.) influence the set of mentioned colors?

https://theeyetribe.com/
http://press.liacs.nl/mirflickr/

D. Procedures

The sessions took place in a room properly prepared for the study, with adequate lighting and isolation from external interferences. We started the study by showing to the users three plates (4, 7 and 17) from the Ishihara 24 plates test [19], to check for color blindness. Participants who did not pass the test were discarded.

For those who passed the test, we started by collecting demographic information about them, namely age, gender and whether they were wearing glasses or contact lenses, and calibrated the eye tracker. Then, we presented 100 images to each user, one at a time, during seven seconds. For each image users verbally enumerated the names of the colors, while our application registered the coordinates of the image at which they looked using the eye tracker.

Half of the users (G1) enumerated up to three colors that they consider to be the dominant ones, while the other half (G2) enumerated up to three colors that they would use if they wanted to search for the image. The names of the colors were not defined a priori, so users could say any name they wanted. We registered those names as users enumerated them.

IV. Results

This section presents the main results from our study and answers our research questions. Finally, we describe the resulting dataset containing the images, their dominant and search colors, and the gaze coordinates collected.

A. Color Names

After collecting the color names and the gaze coordinates for each user and image, our first step was to group the names of the colors mentioned by users, to see if we could reduce them to a small palette. We performed this separately for each group (G1 - dominant colors, G2 - search colors).

From the analysis of the names, we found that they could be grouped into a reduced number of colors. In fact, the names mentioned more often by the users were the 11 JNS colors, defined by Berlin and Kay. Table II presents the colors enumerated by the participants and how we grouped them into the 11 colors palette. As we can see, for each color of the palette, the color most mentioned was equal to that of the palette. In fact, 90.7% (dominant colors) and 94.0% (search colors) of the names mentioned by the users belonged to the 11 colors palette. These results are in line with our previous study [14], where we found an agreement of 94.6%.

From this, we can conclude that the 11 JNS color palette is appropriated for the identification of dominant colors and the specification of colors for searching. Furthermore, it contains colors whose names people can easily enumerate, enabling them to specify colors using various modalities, such as speech, writing or sketches, making the creation of queries for content-based retrieval or color exploration systems more natural, easier, and simpler to perform.

We could have used the palette introduced by Ware in the scope of an application for nominal information coding [20, p. 126], which is composed of the 11 JNS colors plus the cyan,

Table II
COLORS ENUMERATED BY THE PARTICIPANTS AND HOW WE GROUPED THEM INTO THE 11 COLORS PALETTE.

Color	Dominant Colors (G1)			Search Colors (G2)		
Palette	Total	#	Names	Total	#	Names
White	942	935	White	993	992	White
		3	Off-White		1	White Light
		3	White Light			
		1	Transparent White			
Black	520	519	Black	555	554	Black
		1	Ebon		1	Black Gray
Gray	336	320	Gray	290	284	Gray
		7	Light Gray		3	Light Gray
		5	Dark Gray		1	Grayish Brown
		1	Cement		1	Gray Cream
		1	Gray-medium		1	Silver
		1	Gray Tree			
		1	Silver			
Red	487	451	Red	507	481	Red
		14	Brick		14	Brick
		12	Bordeaux		6	Bordeaux
		4	Wine		3	Red Pink
		2	Red Pink		1	Dark Red
		1	Red Brown		1	Wine
		1	Red-sly		1	Garnet
		1	Red wine			
		1	Vermilion			
Brown	724	497	Brown	687	524	Brown
		59	Beige		51	Beige
		59	Skin color		49	Skin Color
		46	Cream		23	Cream
		25	Light Brown		20	Light Brown
		21	Dark Brown		11	Dark Brown
		2	Cream Brown		2	Beige Yellow
		2	Sepia		1	Yellowish Brown
		1	Light Beige		1	Reddish Brown
		1	Dark Beige		1	Brown Earth
		1	Camel		1	Skin Brown
		1	Yellowish Brown		1	Cream Brownish
		1	Brown Beige		1	Maroon
		1	Camel Brown		1	Honey
		1	Gray-brown			
		1	Greenish Brown			
		1	Dirty Brown			
		1	Brown Earth			
		1	Brownish Brown			
		1	Brown Tree			
		1	Creamy			
Orange	170	160	Orange	172	166	Orange
		1	Reddish orange		5	Redhead
		1	Orange Brick		1	Reddish orange
		1	Peach			
		7	Redhead			
Yellow	561	525	Yellow	582	566	Yellow
		13	Golden		6	Golden
		5	Light Yellow		5	Sand Yellow
		5	Yellow Roasted		2	Diarrhea Yellow
		4	Ocher		1	Yellow Yellow
		3	Dark Yellow		1	Sand Yellow
		3	Blond		1	Earth Yellow
		2	Yellowish			
		1	Yellow Vomit			
Green	898	836	Green	959	937	Green
		29	Dark Green		10	Dark Green
		8	Light Green		4	Light Green
		6	Lettuce Green		3	Greenish Yellow
		5	Forest Green		1	Vegetation Green
		4	Acid Green		1	Greenish Brown
		2	Greenish Yellow		1	Greenish Blue
		2	Green Petroleum		1	Grass Green
		2	Greenish		1	Aqua Green
		1	Olive Green			
		1	Greenish Blue			
		1	Lime Green			
		1	Pale Green			
Blue	803	707	Blue	853	794	Blue
		31	Dark Blue		20	Dark Blue
		27	Turquoise		16	Light Blue
		6	Aquamarine		8	Turquoise
		5	Light Blue		4	Sea Blue
		5	Indigo Blue		3	Cyan
		5	Sea Blue		1	Greyish Blue
		5	Navy Blue		1	Dark Blue Gray
		2	Cobalt Blue		1	Navy Blue
		2	Blue Baby		1	Night Blue
		2	Blue Cyan		1	Sky Blue
		2	Blue Green			
		2	Sky Blue			
		1	Greyish Blue			
		1	Blue Gray			
Purple	94	69	Purple	105	64	Purple
		14	Lilac		24	Lilac
		11	Violet		15	Violet
					2	Light Purple
Pink	116	106	Pink	136	128	Pink
		4	Magenta		5	Magenta
		1	Pink Skin Color		2	Hot Pink
		1	Pink Bordeaux		1	Pink Skin Color
		1	Light pink			
		1	Pink Fluorescent			
		1	Salmon			
		1	Fuchsia			

but from our analysis people mentioned cyan a very reduced number of times (only twice for dominants and three times for search). Thus, and despite this 12 colors palette being used by Google and Bing in their image search engines, we found that the 11 JNS colors palette is more natural to users.

B. Dominant Colors vs Search Colors

One of our research questions (RQ_2) seeks to know whether the colors that people use to search for an image are related to the dominant colors of that image. To that end, we started by identifying the most voted colors for each image and for each situation (dominant and search).

We consider a color to be a dominant or search color for an image if it has more than 10% of the votes for that image. We defined this threshold based on our previous tests, where we found that a color with less than 10% has a very low importance on an image [14].

With this approach, we could assign more than the three colors that we asked users to mention, i.e., we decided not to limit the number of colors to three because: 1) some colors can have the same percentage of votes, and we should not ignore one of them just because there are more than three colors; and 2) people perceive colors differently, e.g., some shades of red can be perceived as orange or as brown. Thus, if a significant amount of people identify that in a specific image the existing reds are "brown" or "orange", this should be reflected on the colors that describe the image. As an example, consider an image that has the following distribution of votes: 35% black, 24% red, 14% white, 14% yellow, 6% blue, 4% orange, and 3% gray. The resulting set of colors will be black, red, white, and yellow, since they have more than 10% of the votes.

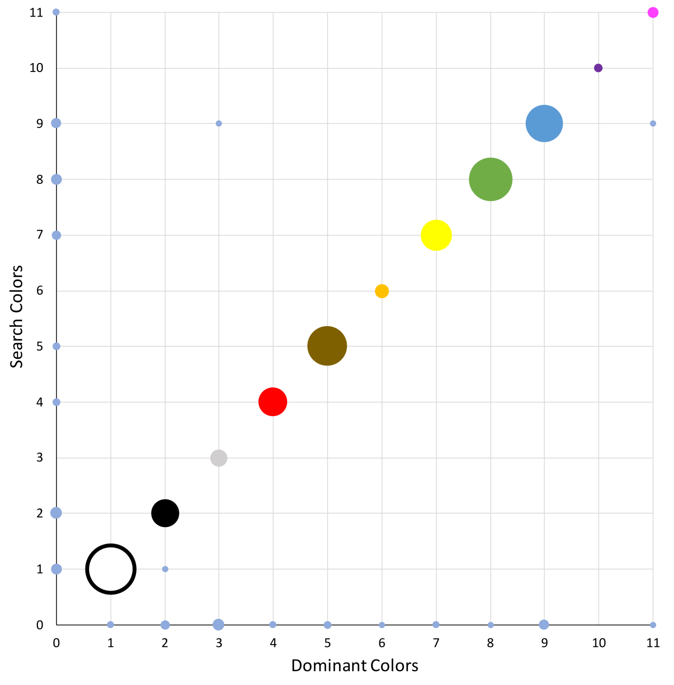


Figure 1. Distribution of the votes for all the images across the eleven colors: white (1), black (2), gray (3), red (4), brown (5), orange (6), yellow (7), green(8), blue (9), purple (10), and pink (11). Zero represents a color that was used as dominant/search but was not used for search/dominant.

After assigning the most voted colors (dominant and search) to all images, we aligned the similar dominant and search colors for each image. We ended up with a set of 400 pairs, some composed of two colors that are similar on both sides (e.g. green-green) and others where we have only one color on one of the sides (e.g. green-none, or none-green). The latter means that there was no similar color on the other side.

Figure 1 presents the distribution of these pairs across the eleven colors. In the diagonal, we can see the colors that were used simultaneously as dominant and for search, while the size of the bubble represents the amount of times that this pair occurred. We have a correspondence of 80.5% between the dominant colors and the search colors, 0.75% where the two colors are different, 10.25% where we have a color for search but not for dominant, and 8.50% on the opposite case.

To assess the agreement between the dominant and search colors, we used similarity metrics to quantify how similar two sets of colors are (dominant colors are denoted by D, while search colors are denoted by S). The measures used were the *Jaccard index* [21] (see Eq. 1), the *Sørensen-Dice index* [22], [23] (see Eq. 2), and the *Overlap coefficient* [24] (see Eq. 3). For all these metrics, the closer its value is to one (or 100%), the more similar the two sets are.

$$jaccard(D,S) = \frac{|D \cap S|}{|D| + |S| - |D \cap S|} \tag{1}$$

$$sorensenDice(D,S) = \frac{2\,|D \cap S|}{|D| + |S|} \tag{2}$$

$$overlap(D,S) = \frac{|D \cap S|}{min\Big(|D|, |S|\Big)} \tag{3}$$

Let us consider the following example: we have an image with dominant colors white, red, and green, while the search ones are white, red, green, and blue. White, red, and green colors are common to dominant and search, but blue is not.

If we we are concerned with exact matches, we should use the $jaccard$ or $sorensenDice$ to assess the agreement. In such case, we would have an agreement of 75% for $jaccard$ and 86% for $sorensenDice$, i.e., in both cases we would be penalizing the result due to the existence of an extra color (blue). Otherwise, we should used $overlap$ that will only consider the exact matches, even if there are more colors assigned to dominant than search, or vice-versa. In this case, we would have an agreement of 100%.

Table III presents a summary of our dataset. We present the number of images per category, the average number of dominant colors and search colors assigned to each category, and the average agreement percentage for each similarity metric. As we can see, around half of the categories (53.44%) have the same average for dominant and search colors, while 33.33% have an average of search colors bigger than the dominant.

For the dominant colors, the following categories have at

Table III
OVERVIEW OF OUR DATASET, SHOWING THE NUMBER OF IMAGES PER CATEGORY, THE AVERAGE NUMBER OF COLORS PER CATEGORY AND THE AVERAGE VALUES FOR EACH METRICS.

Category	#	AvgDC	AvgSC	jaccard	sorensenDice	overlap
Animal	4	3.75	4.00	72.5	82.8	85.5
Architecture	3	3.67	3.67	85.0	92.7	100
Baby	3	4.00	4.00	70.0	81.7	89.0
Beach	4	3.20	3.60	71.3	86.0	100
Bird	4	3.50	3.75	85.5	91.5	100
Building	3	3.00	3.33	75.0	84.3	89.0
Car	4	3.25	3.75	87.5	93.0	100
Clouds	2	3.00	3.00	75.0	83.5	83.5
Dog	3	4.00	4.00	86.7	92.7	100
Flowers	4	3.50	3.50	75.0	84.8	100
Food	3	3.67	3.67	85.0	91.7	100
Girl	4	3.25	3.75	88.8	93.8	100
Graffiti	4	4.00	3.50	76.3	87.5	100
Lake	3	4.00	4.00	89.0	93.3	93.3
Landscape	4	2.75	3.00	79.3	86.8	91.8
Nature	3	3.67	3.67	75.0	84.3	100
Night	3	4.00	3.67	78.3	87.0	91.7
People	4	3.50	3.50	75.0	84.8	91.8
Portrait	3	4.67	4.00	74.0	85.0	100
River	3	4.00	4.00	100	100	100
Sea	3	3.67	3.67	83.3	89.0	89.0
Sign	3	4.00	3.67	91.7	95.3	93.8
Sky	4	3.50	3.50	90.0	93.8	93.8
Snow	3	3.00	3.00	90.0	93.8	93.8
Street	3	4.33	4.33	63.3	77.0	83.3
Sun	3	3.33	3.33	83.3	90.7	100
Sunset	3	3.33	3.67	91.7	95.3	100
Trees	3	3.33	3.33	83.3	90.7	100
Urban	4	3.25	4.00	83.8	90.3	100
Water	3	4.00	3.67	93.3	96.3	100
Total	100	3.59	3.66	82.1	89.3	96.2

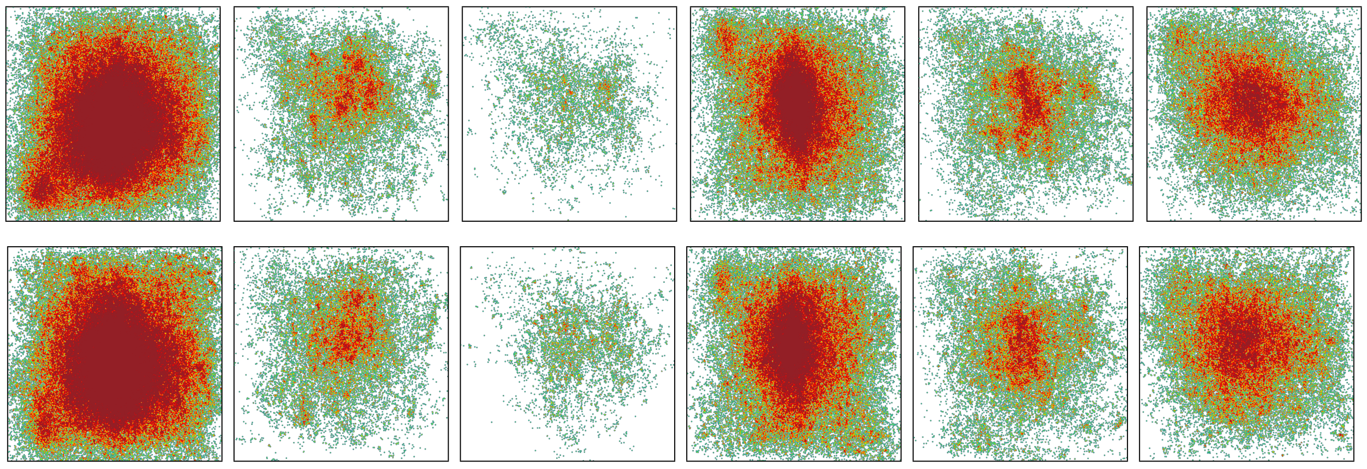


Figure 2. First row depicts the heatmaps for the dominant colors, and the second row for the search colors. At a given position, a darker shadow of red represents a stronger number of eye gazes, yellow and green represent a medium number, and blue a lower number. (a) all the categories; (b) animal, bird, and dog categories; (c) food category; (d) beach, clouds, flowers, landscape, lake, nature, river, sea, sky, snow, sun, sunset, trees, and water; categories; (e) baby, girl, people, and portrait categories; (f) architecture, buildings, car, graffiti, night, sign, street, and urban categories. (best seen in color)

least an average of four dominant colors: portrait, street, baby, dog, graffiti, lake, night, river, sign, and water; while building, clouds, snow, and landscape categories have three or less dominant colors. Regarding the search colors, the following categories have at least an average of four colors: street, animal, baby, dog, lake, portrait, river, and urban, while clouds, snow, and landscape categories have three or less colors.

If we analyze our results considering the most restrictive measures, we have a $jaccard$ agreement varying from 72% to 100%, and a $sorensenDice$ agreement varying from 82.23% to 100%. The most permissive of the three measures, the $overlap$ varies from 89% to 100%. If we now consider the overall dataset, we have an average agreement of 82.12%±17.04% using $jaccard$, 89.28%±10.80% using $sorensenDice$, and 96.22%±9.99% using $overlap$.

From these values, we can conclude that there is virtually no difference for users when asked about dominant colors in an image and colors to be used for searching for that images. In conclusion, a possible algorithm that identifies dominant colors in images according to human perception, will also serve to highlight the colors that would be used by a user to search for the same image.

C. Focus Regions

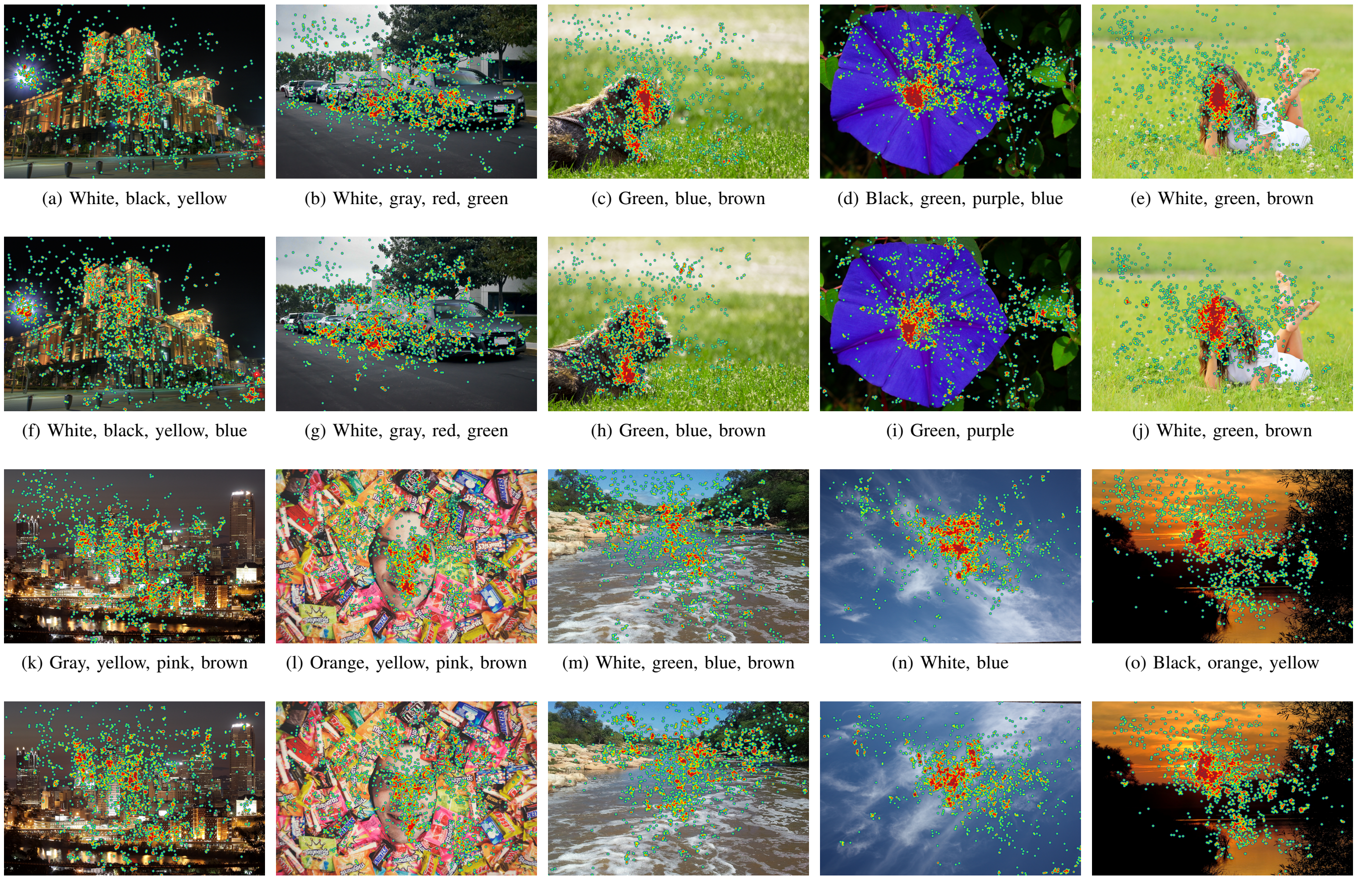
Before we analyzed the gaze information, we validated for each participant if there were any corrupted data to be removed (e.g., coordinates outside the image). Across all the images and participants, we had a total of 287 538 gaze coordinates for the dominant colors and 268 716 for the search colors. We discarded around 7% of corrupted data from the former and around 11% from the latter.

To analyze and identify the gaze patterns, we created heatmaps for each image, groups of categories, and the overall dataset, considering the dominant and search colors separated. Since we have images with different orientations and sizes, we normalized the gaze coordinates for each image according to their max width and height. This way, we ensure that our conclusions are correct regardless of the orientation and size of the images. Figure 2 presents the normalized heatmaps of our dataset for both dominant and search colors. To simplify the analysis, we created groups of categories by joining related ones (e.g. animal, bird, and dog). We can see that people look at the central area of images, regardless of being questioned about dominant or search colors (Figure 2a). This is also true for the different groups of categories (Figures 2b - 2f).

Figure 3 present examples of images from our dataset with the corresponding heatmaps overlapped, and the dominant and search colors associated to each one.

Figures 3a and 3f depict a building illuminated at night. People looked more at the center of the image, where we can find the main part of the building, the lamp light and the red lights of traffic. The white, black and yellow colors reflect this gaze behavior, but black (for dominant and search) and blue (for search) are not predominant in the areas where people looked. Figures 3b and 3g depict a street with parked cars. Although, people identified white (surroundings and buildings), gray (car on front and street), red (car), and green (trees) as the dominant and search colors, in both cases, they mainly looked at the red car.

Figures 3c and 3h show a dog resting on grass. In this case, people mainly looked at the dog face and dog-collar. The predominant colors were green (grass), blue (dog-collar), and finally brown (dog body and face). It is interesting to notice that regardless of the small size of the dog-collar (when compared with the size of the dog), the blue color had more votes than the brown. Figures 3d and 3i depict a purple flower. Here, people mainly looked at the stigma of the flower (white/yellow area in the middle of the flower), the top part of the flower, and some leaves. The identified search colors were purple and green (flower), while for dominant colors, the black and blue colors were also identified.



(a) White, black, yellow (b) White, gray, red, green (c) Green, blue, brown (d) Black, green, purple, blue (e) White, green, brown

(f) White, black, yellow, blue (g) White, gray, red, green (h) Green, blue, brown (i) Green, purple (j) White, green, brown

(k) Gray, yellow, pink, brown (l) Orange, yellow, pink, brown (m) White, green, blue, brown (n) White, blue (o) Black, orange, yellow

(p) White, gray, yellow, brown (q) White, black, yellow, blue (r) White, green, blue, brown (s) White, blue (t) Black, orange, yellow

Figure 3. Examples of heatmaps and images for the dominant colors (first and third row) and search colors (second and forth row). (best seen in color)

Figures 3e and 3j show a young girl laying on the grass. We can see that people mainly looked at the girl face, but indicated white (dress), green (grass), and brown (hair and maybe skin) as the predominant colors for both search and dominant. Figures 3k and 3p depict a nightscape with buildings across the river. Similarly to Figures 3a and 3f, people mainly looked at the center of the image where the buildings and lights are concentrated. For this image, the dominant colors were gray (from the sky and maybe buildings), yellow (from the buildings lights), pink (maybe the central building resembles light pink, and the top structure at its left, dark pink), and brown (surroundings and shadows). It is interesting that in search colors, people also looked at the top of the building with a white light (right top part of image) and the building front illuminated with a white light (right middle part of image). As a result, white was one of the predominant colors identified.

In Figures 3l and 3q, we have the face of a man surrounded by packages of chocolates. In both cases, people mainly looked at his face. However, in both cases, people identified the colors of the chocolate packages (e.g., orange, yellow, white, black). Figures 3m and 3r depict a river with some vegetation. People looked more at the top of the image, where the vegetation and the narrowest river area are. In both cases, people identified white (from the water foam), green (vegetation), blue (from the narrowest part of the river), and brown (from the banks and wider area of the river) as the predominant colors.

In Figures 3n and 3s, we have the sky with clouds. People mainly looked at the center of the image, where the biggest portion of the clouds are. Not surprisingly, the predominant colors identified were white and blue. Finally, in Figures 3o and 3t, we have a sunset on the river. People looked to the sun and the area around it. However, the most predominant color was black, where people barely looked at.

D. Discussion

Based on the results from our study, we will answer now the research questions that we raised in Section III.

According to Table II, we can say that the answer to our RQ_1 is yes, that is, we can reduce all the color names mentioned by users to a small subset of colors, such as the 11 colors palette suggested by Berlin and Kay. From the comparison and the assessment that we made on Section IV-B, we verified that there is a strong similarity between the dominant colors and search colors mentioned for each image. Thus, we can say that the colors that users would use to search for an image are the dominant colors of the image (RQ_2).

Although, the gaze pattern differs a bit among the groups of categories (RQ_5), as illustrated in Figure 2, the most looked region is the center of images (RQ_3). Moreover, we noticed that people do not look at some regions of the image, but enumerate their colors, and look at other parts of the image (e.g. faces, bright spots, lights) and do not mention their colors (RQ_4). We noticed that people identify as predominant colors, colors from small areas of the image probably because they have striking colors (e.g., red car, blue dog-collar) (RQ_6).

In summary, we can say that users mentioned colors from the whole image and not only from the area where they looked at. In particular, we noticed that users focus on faces, but identify as predominant colors those of the surrounding objects (e.g. hair, clothes). This focus on faces was also observed by Cerf *et al.* in their study [25]. Finally, and although people use the same "scanning" method for the identification of the dominant and search colors, they slightly tend to disperse more their gaze while identifying colors for searching purposes.

E. Resulting Dataset

Our dataset, named UL-GDSC (Gaze on Dominant and Search Colors), is composed of 100 images collected from Flickr and resized to match the largest size of the screen (width of 1600 or height of 900 pixels). Images are organized in 30 categories, as shown in Table III, and are annotated with their dominant colors, the colors that people would use to search for them, and the coordinates where people gaze at while identifying the colors. We made UL-GDSC dataset publicly available to the community.

Each image has two sets of colors (dominant and search colors) based on the colors that received more than 10% of the votes. On average, images have three to four colors associated. The gaze coordinates in the dataset are the average of three consecutive raw coordinates provided by the eye tracker. Thus, we were able to have more stabilized gaze coordinates, with the cost of having less values per second, since we indirectly reduced the sampling rate (from 60 Hz to 20 Hz).

A salient aspect of the UL-GDSC is that it contains not only the colors that people identified as dominant and for searching, but also the eye movements people performed while doing it.

V. Conclusion

In this paper, we presented the results of a study with users to identify the predominant colors in images and at which regions they look while mentioning those colors. From the data collected, we were able to confirm that the JNS palette contains a set of colors that is representative of the color names that users mentioned.

Additionally, we measured the similarity between the dominant colors associated to an image and the colors used to search for it, and found that they are very similar. So, we can use the dominant colors of the images as a content descriptor, since users would use them for searching.

The analysis of the gaze data revealed that overall there is no strong relation between the colors of the regions where people look at and the predominant colors identified in the image. Furthermore, people look mainly at the center of the image, regardless of its category.

http://www.di.fc.ul.pt/~mjf/research/ul-gdsc/

Acknowledgment

This work was supported by national funds through Fundação para a Ciência e Tecnologia, under LASIGE Strategic Project - UID/CEC/00408/2013.

References

[1] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive Psychology*, vol. 12, no. 1, p. 97136, Jan. 1980.

[2] B. Berlin and P. Kay, *Basic color terms : their universality and evolution*. University of California Press, 1969.

[3] A. Mojsilovic, "A computational model for color naming and describing color composition of images," *Transactions on Image Processing*, pp. 690–699, 2005.

[4] J. van de Weijer, C. Schmid, and J. Verbeek, "Learning color names from real-world images," in *Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[5] N. Moroney, "Unconstrained web-based color naming experiment," in *Color imaging VIII: Processing, hardcopy, and applications*. International Society for Optics and Photonics, 2003, pp. 36–47.

[6] G. Menegaz, A. Le Troter, J. Sequeira, and J.-M. Boi, "A discrete model for color naming," *Journal on Advances in Signal Processing*, 2007.

[7] R. Benavente, M. Vanrell, and R. Baldrich, "Parametric fuzzy sets for automatic color naming," *Journal of the Optical Society of America A*, pp. 2582–2593, 2008.

[8] E. Y. Chang, B. Li, and C. Li, "Toward perception-based image retrieval," in *Workshop on Content-based Access of Image and Video Libraries*, 2000, pp. 101–105.

[9] J. R. Smith and S.-F. Chang, "Visualseek: a fully automated content-based image query system," in *ACM international conference on Multimedia*, 1997, pp. 87–98.

[10] Y. Deng, B. Manjunath, C. Kenney, M. S. Moore, and H. Shin, "An efficient color representation for image retrieval," *Transactions on Image Processing*, pp. 140–147, 2001.

[11] A. Mojsilovic, H. Hu, and E. Soljanin, "Extraction of perceptually important colors and similarity measurement for image matching, retrieval and analysis," *Transactions on Image Processing*, pp. 1238–1248, 2002.

[12] A. Atsalakis and N. Papamarkos, "Color reduction and estimation of the number of dominant colors by using a self-growing and self-organized neural gas," *Engineering Applications of Artificial Intelligence*, 2006.

[13] A. A. Younes, I. Truck, and H. Akdag, "Image retrieval using fuzzy representation of colors," *Soft Computing*, pp. 287–298, 2007.

[14] J. C. Amante and M. J. Fonseca, "Fuzzy color space segmentation to identify the same dominant colors as users," in *International Conference on Distributed Multimedia Systems*, 2012, pp. 48–53.

[15] A. Talib, M. Mahmuddin, H. Husni, and G. Loay E, "A weighted dominant color descriptor for content-based image retrieva," *Journal of Visual Communication and Image Representation*, pp. 345–360, 2013.

[16] J. Lazar, J. H. Feng, and H. Hochheiser, *Research Methods in Human-Computer Interaction*. John Wiley & Sons, 2010.

[17] S. Winkler, F. M. Savoy, and R. Subramanian, "X-eye: A reference format for eye tracking data to facilitate analyses across databases," in *Human Vision and Electronic Imaging*, 2014.

[18] S. Winkler and R. Subramanian, "Overview of eye tracking datasets," in *International Workshop on Quality of Multimedia Experience*, 2013.

[19] S. Ishihara, *Tests for color-blindness*. Handaya, Tokyo, Hongo Harukicho, 1917.

[20] C. Ware, *Information Visualization: Perception for Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[21] M. J. Zaki and W. M. Jr, *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2014.

[22] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology*, pp. 297–302, 1945.

[23] T. A. Sørensen, “A method of establishing groups of equal amplitude in plant sociology based on similarity of species content, and its application to analyses of the vegetation on {Danish} commons,” *Biologiske Skrifter*, pp. 1–34, 1948.

[24] Y. A. Pesenko, “Principles and Methods of Quantitative Analysis in Faunistical Researches,” *Moscow (Nauka) [in Russian]*, 1982.

[25] M. Cerf, J. Harel, W. Einhaeuser, and C. Koch, “Predicting human gaze using low-level saliency combined with face detection,” in *Advances in Neural Information Processing Systems*, 2008, pp. 241–248.

Enriching Image Datasets with Unrestrained Emotional Data: A Study with Users

Soraia M. Alarcão and Manuel J. Fonseca
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
salarcao@lasige.di.fc.ul.pt, mjfonseca@ciencias.ulisboa.pt

***Abstract*—Elicitation of emotions is typically done through the presentation of emotionally salient material, like images or videos, thus requiring reliably annotated datasets. Although there are datasets with emotional information, these only describe either emotional polarities or discrete emotions. The only available dataset with both types of information restrained the participants during the study by separating a priori the images according to their polarity (positive or negative). In this paper, we describe an unrestrained study with 60 participants, where we asked them to rate the polarities and discrete emotions elicited by a set of images. The analysis of the emotional ratings made by the users revealed the most frequent correlations between the basic emotions. Furthermore, the analysis of the ratings' agreement among participants and existing datasets shows that our results are aligned with the existing ones. As a result of our study, we make available to researchers a more informative picture dataset annotated with emotional polarities and multiple emotions, as a complement to existing datasets.**

I. Introduction

The role of emotions in human cognition is essential given their importance in the daily life of human beings. Emotions play a critical role in rational decision-making, perception, human interaction, and intelligence [1], [2].

In the last decade, there has been an increasing body of work involving emotions: to improve content-based classification for both music and video, using photos and emotions conveyed by multimedia [3]; to gather emotional information from images through their visual content [4]; to observe the emotional state of a person using Electroencephalography [5]; to improve interactive experiences using user emotional expressions [6]; and finally, to enhance the quality of recommendation systems [7].

Besides these examples, many studies in psychology and computer science involve manipulating emotions via emotional stimuli [8]. If a stimulus is relevant enough, an appraisal is automatically executed and will trigger reactions in measurable components of emotion, such as physiological responses, expressivity, action tendencies, and subjective feelings. Several methods have been introduced for priming participants, such as the presentation of emotionally salient material like images [9], audio [10], video [11], or text [12]. The use of the visual channel remains the most common to convey emotional stimulation [13].

DOI reference number: 10.18293/VLSS2018-033

In the different areas of research based on visual stimulation, reliable datasets are important for the success of emotion induction. To that end, in 1997, the International Affective Picture System (IAPS) dataset was presented [14]. Later, in 2011 and 2014, two new datasets were created: Geneva Affective PicturE Database (GAPED) [13], and Nencki Affective Picture System (NAPS) [15]. These increased the availability of visual emotion stimuli, while trying to solve the problem of a limited number of pictures for specific themes. IAPS only provides valence and arousal, while GAPED has some information about the emotional polarity (negative, neutral or positive) of their images, but it is not enough for the cases where there is the need to use discrete emotions.

To minor the lack of emotional information, in 2005 and 2016, Mikels [16]–[18] and NAPS Basic Emotions (NAPS-BE) [19] were presented. Mikels collected descriptive emotional data on a subset of the IAPS to identify the elicited discrete emotions. Although this work enriched the emotional information associated to the IAPS dataset, we believe that the authors have restrained the choices of the participants by asking them to select discrete emotions only in a specific polarity (positive or negative), according to the subset where the image was placed a priori by the authors. This restriction prevented mixtures of positive and negative emotions. However, it is possible that an image arouses positive emotions in a person and negative in another. Finally, authors did not consider that images could be neutral.

In this paper, we present a study about the experience of viewing a set of images from the IAPS and GAPED datasets. We focused on the process of rating the images according to the emotions and polarities they elicited in the viewer, as well as the participants' insights during the experience. Although it would be interesting to use images from the NAPS-BE, it was not yet available when we conducted the study. Our contributions are: 1) a more complete and realistic picture dataset composed of 169 images, each annotated with information about the predominant emotional polarity, the intensity of each discrete emotion elicited by the image, and the valence and arousal values from the original datasets; 2) the relationship between multiple emotions that arise when visualizing images, that are in line with the literature, thus confirming the quality of our dataset emotional annotation; 3) our experimental procedure designed to provide more comfort to the users, avoiding stress and fatigue.

II. Background and Related Work

In this section, we briefly explain what are emotions and how we can represent them. We also describe the most commonly used datasets of images to elicit emotions.

A. Emotions

Polarity provides a coarse indication of the emotional image content (positive, neutral, and negative). Emotions, on the contrary, give a more detailed description of the emotional information conveyed. These have been described as discrete and consistent responses to external or internal events with particular significance for the human organism [20]. This finer distinction of emotions provides a richer emotional classification, making it suitable for specific research purposes, such as studying the neuroanatomical correlations among basic emotions when a person is exposed to multimedia stimuli [21].

When talking about emotions, it is important to mention the subjectivity inherent, since multiple emotions can appear in the same subject while looking, for example, at a picture, as well as different subjects can feel different emotions when viewing the same picture, mainly due to each subject's current emotional state and "life experiences" [22], [23]. However, the expected affective response can be considered objective, as it reflects the more-or-less unanimous response of a general audience to a given stimulus [24].

Regarding the existence of multiple emotions while viewing an image, these correlations of basic emotions are a well-known phenomena in the field of psychology. One of the most important results was that when happiness rises, all other emotions decline; another one is that fear correlates positively with sadness and anger [25], [26].

B. Emotions Representation

There are two different perspectives towards emotion representation: categorical and dimensional. The first indicates that basic emotions have evolved through natural selection. Plutchik proposed eight basic emotions (acceptance, anger, curiosity, disgust, fear, joy, sadness, and surprise), from which we can define all the others [27]. Ekman based his work in the relationship between facial expressions and emotions derived from the universal basic emotions (anger, disgust, fear, happiness, sadness, and surprise) [28]. These emotions are considered universal since their external manifestation seems to be independent of culture and personal experiences [29].

In the dimensional perspective, which is based on cognition, the emotions are mapped into the Valence, Arousal and Dominance (VAD) dimensions. Valence goes from unpleasant to pleasant, arousal goes from states like sleepy to excited, and finally, dominance corresponds to the strength of the emotion [14], [30]. The most common model used is the Circumplex Model of Affect (CMA), where all affective states arise from cognitive interpretations of core neural sensations that are the product of valence and arousal [31].

In this work, we used Ekman's set of universal emotions (anger, disgust, fear, happiness, sadness, and surprise) complemented with the neutral emotion.

TABLE I
Comparision among the most commonly used datasets of images.

Dataset	#Images	V-A	Polarities	Emotions
IAPS	1182	Yes	No	No
EmoPics	378	Yes	No	No
GAPED	730	Yes	Yes	No
NAPS	1356	Yes	No	No
POFA	110	No	No	Yes
KDEF	4900	No	No	Yes
NimStim	646	No	No	Yes
ArtPhoto	807	No	No	Yes
Abstract	228	No	No	Yes
Mikels	330	Yes	Yes[2]	Yes
NAPS-BE	510	Yes	No	Yes

C. Image Datasets

In all the different areas of research based on visual stimulation, reliable databases are important for the success of emotion induction. In Tables I and II, we briefly present the most commonly used datasets of images to elicit emotions.

As we can see in Table I, only GAPED and Mikels provide information about the polarity of an emotion, i.e., negative, neutral or positive (Mikels does not consider the neutral polarity). In Mikels, the authors defined the emotional polarity of an image before the participants performed their rating about the discrete emotions. Given the subjectivity inherent to emotions, this could have restrained the results since it did not allow people to express positive emotions for "negative" images, and vice-versa. For example, Yoon *et al.* concluded that some of images did not have agreement between the tags assigned by the image creators and the ones given by image viewers [32].

Machajdik datasets (Art Photo and Abstract Paintings) [33], Mikels, and NAPS-BE discriminate the emotions elicited by images. However, Abstract Paintings is focused in a very specific type of images that are not usually found in personal collections, while the ratings for images of the Art Photo were only done by the artists. IAPS, Emotional Picture Set (EmoPicS) [34], and NAPS do not provide any information about the emotional content of their images, offering only valence and arousal information or physical characteristics of the images. Karolinska Directed Emotional Faces (KDEF) [35], NimStim Face Stimulus Set (NimStim) [36], and Pictures of Facial Affect (POFA)[1] were only labeled with facial expressions and corresponding emotions.

Some datasets have Valence and Arousal (VA) information, but no emotional data; others have emotional information, but no VA; and finally, only GAPED, NAPS-BE, and Mikels have both, but they are restrained and limited.

[1]http://www.paulekman.com/product/pictures-of-facial-affect-pofa/

[2]The emotional polarity (negative or positive) for each image was defined by the authors, not collected from the participants.

TABLE II
DESCRIPTION OF THE MOST COMMONLY USED DATASETS OF IMAGES TO ELICIT EMOTIONS.

Dataset	Description
IAPS	It contains 1182 images, and provides a set of normative emotional stimuli for experimental investigations of emotion and attention. The authors rely on a dimensional view, in which emotions are defined by a coincidence of values on a number of VAD dimensions. Each picture is characterized in terms of their valence and arousal ratings. They were made by males, females and children using Self-Assessment Manikin (SAM) questionnaires during 10 years [37].
EmoPicS	It contains 378 standardized color images with different semantic contents, such as social situations, animals, and plants, selected from public online photo libraries and archives. Each image of the database was rated with their corresponding dimensional information: valence and arousal, and also with some physical characteristics of the given image: color composition, contrast, and luminance.
GAPED	It contains 730 pictures: 121 representing positive emotions using human and animal babies as well as natural sceneries, 89 for the neutral, mainly using inanimate objects, and 520 for the negative, using spiders, snakes, human rights violation, and animal mistreatment. The pictures were rated according to valence, arousal, and the congruence of the represented scene with moral and legal norms regarding Swiss legislation, since the study was conducted in Switzerland. These ratings were made by 60 subjects, where each subject rated 182 images.
NAPS	It contains 1356 realistic, high-quality images divided into five categories: animals, faces, landscapes, objects, and people. Besides valence, arousal and motivational direction (avoidance-approach) ratings, each image was annotated with some physical characteristics, namely color composition, contrast, and luminance. 204 subjects made the ratings, where each one rated 362 images, pseudo-randomly chosen from all the categories with the constraint that no more than three stimuli of the same category were presented in succession.
POFA	This dataset consists of 110 photographs of facial expressions that have been widely used in cross-cultural studies, and more recently, in neuropsychological research. All images are in black and white, and each image has a set of norms associated. It is important to note that the images are not identical in intensity or facial configuration.
KDEF	It is a set of 4900 pictures of human facial expressions of emotion suitable for perception, attention, emotion, and memory. Thus, special attention was given to photograph expressions at different angles, with soft light, and using t-shirts with uniform colors. A grid was used to center the face of the users during shooting, as well as position the eyes and mouth in certain coordinates of the image during scanning. The set contains 70 individuals, each displaying seven different emotional expressions, which were photographed from five different angles.
NimStim	It consists of 646 facial expression stimuli. Images include fearful, happy, sad, angry, surprised, calm, neutral, and disgusted expressions displayed by a variety of models of various genders and races. Examples of facial expressions were shown to the actors, for them to get an idea of what was the aim, and then they posed for each facial expression. Muscles were adjusted until the desired expression was achieved.
Art Photo	It contains 807 artistic photographs that were obtained by using the emotion label as search terms in the deviantArt site. The emotion label was determined by the artist who uploaded the photo, that was trying to evoke a certain emotion in the viewer of the photograph through the conscious manipulation of the image composition, colors, etc.
Abstract	It contains 228 images with combinations of color and texture, without any recognizable objects. To obtain ground truth, images were peer rated in a web-survey where the users could select the emotional category from amusement, anger, awe, contentment, disgust, excitement, fear and sad, for 20 images per session. 230 people rated approximately 280 images, where each image was rated about 14 times.
Mikels	This dataset is composed of 330 images from the IAPS, annotated with positive (amusement, awe, contentment, and excitement) and negative (anger, disgust, fear, and sadness) emotions. Thirty males and 30 females made the emotional category ratings in two studies, using a subset of negative images and a subset of positive images, with a constrained set of categorical labels.
NAPS-BE	This dataset contains 510 images from the NAPS, annotated with the emotions anger, disgust, fear, happiness, sadness, and surprise. It has 98 images depicting animals, 161 faces, 49 landscapes, 102 objects, and 100 people. Sixty seven females and 57 males made the emotional ratings, where each subject rated around 170 images.

III. EMOTIONAL USER STUDY

In this section, we describe the study carried out, in which participants identified both the emotional polarity and emotions they felt while visualizing each image.

A. Participants

Sixty participants completed the study: 26 females and 34 males, with 70% of them belonging to the 18-29 age group, and almost 60% having a BSc Degree. None of the participants had participated in any study using the IAPS or GAPED, and the overwhelming majority had no knowledge about these datasets.

Regarding their emotional state at the beginning of the study, 31 participants classified themselves as neutral, 25 as positive, and only 4 as negative. Considering the discrete emotions (anger, disgust, fear, happiness, neutral, sadness, and surprise), the majority of the participants were feeling moderately happy or moderately neutral, both with a median of 3 in a scale of 1-5, with 1 corresponding to a weak feeling, and 5 to a strong feeling.

B. Apparatus and Material

A MacBook Pro (13-inch) computer was used with an application for participants to see the images and rate the emotions and polarities elicited by each image.

The dataset used in the study was composed of 86 images from the IAPS, 76 images from the GAPED, and 7 images from Mikels' dataset. It contained images with animals (cats, dogs, horses, sharks, snakes, spiders, tigers, among others), car accidents, children, death situations, diseases, fire, mutilation, natural catastrophes, poverty, and war scenarios. We chose a set of images that we believed to represent in a balanced way the discrete emotions throughout the valence-arousal space (see Figure 1).

Since it was impractical and even unpleasant for participants to annotate all the images in our dataset, and also due to the time it would take, we randomly divided our dataset into four subsets: DS0 to DS3. DS0 contained 57 images (30 IAPS, 20 GAPED, 7 Mikels), DS1 contained 40 images (20 IAPS, 20 GAPED), while DS2 and DS3 contained 36 images each (18 IAPS, 18 GAPED).

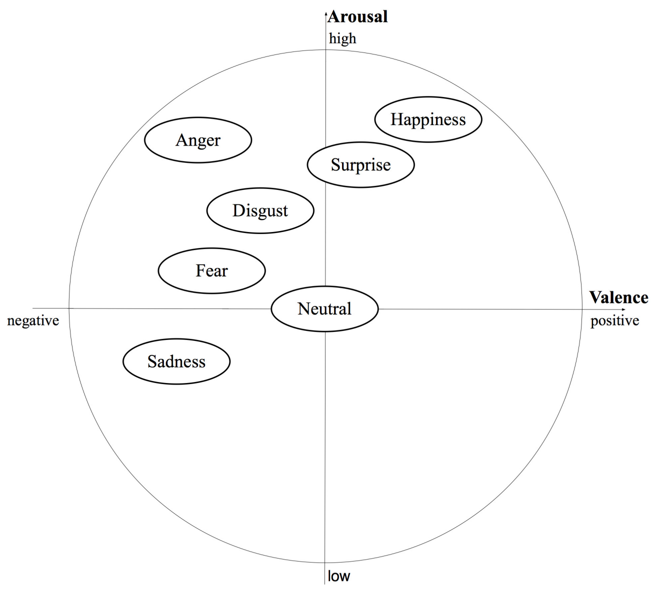


Fig. 1. Adaptation of the Circumplex Model of Affect, mapping the discrete emotions into the Valence-Arousal plane [38].

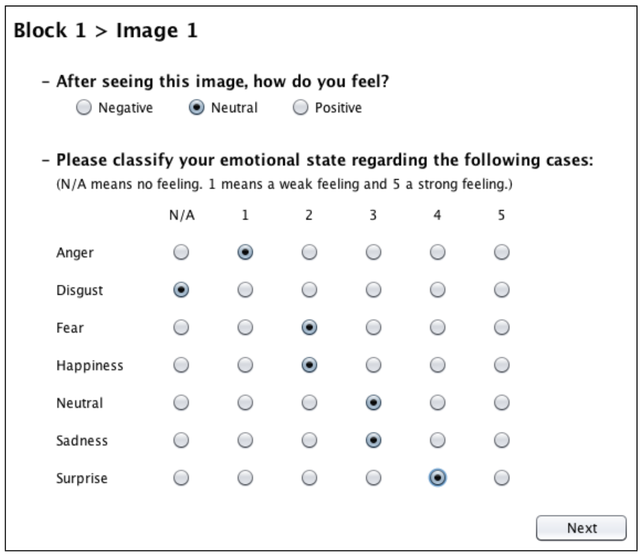
All the participants rated each image of DS0, while images from DS1, DS2 and DS3 were rated by 20 participants. With this process, we managed to get a larger number of annotated images in the shortest time possible.

C. Design and Procedure

The experimental sessions took place in a room properly prepared for the task, aiming at providing comfort to participants, with adequate lighting and isolation from external noises. The option for the solo exhibition seeks to contribute to better control of external interference (e.g., comments from other participants, noise) that could interfere with emotional participant's experience [39].

We started by explaining the purposes of the study and how it would be held. To ensure the willingness of the subjects regarding negative images, we showed three images as examples of what could be expected. After that, the subjects could decide whether to continue or not the study. One participant (not included in the 60) decided not to continue the study due to medical issues. If they accepted, they should fill the participants' questionnaire with their personal information (age, gender, etc.), and the classification of their current emotional state (polarity and emotions).

The first screen of the application presented a summary of the most important aspects of the study. Then, seven blocks of images were presented sequentially, with about 14 images on each block. Each image (with a resolution of 640x480 pixels) was displayed randomly during 5 seconds, and after the visualization, participants evaluated their emotional state (regarding the polarity felt), and rated it for each of the emotions (see Figure 2). To obtain the participants' emotional reactions without practical limitations (e.g. specialized equipment for collecting physiological signals), we adopted a 5-point Likert scale for each emotion.



Block 1 > Image 1

– **After seeing this image, how do you feel?**

Negative [✓ Neutral] Positive

– **Please classify your emotional state regarding the following cases:**
(N/A means no feeling. 1 means a weak feeling and 5 a strong feeling.)

	N/A	1	2	3	4	5
Anger	○	●	○	○	○	○
Disgust	●	○	○	○	○	○
Fear	○	○	●	○	○	○
Happiness	○	○	●	○	○	○
Neutral	○	○	○	●	○	○
Sadness	○	○	○	●	○	○
Surprise	○	○	○	○	●	○

Fig. 2. Rating screen of the application with the 5-point Likert scale.

This process was repeated for each image of the seven blocks of images of our study. Although in similar studies participants usually had a limited time to answer, we decided not to do it. This way, we allowed participants to spend the time they needed, without feeling pressured to respond or even stressed out. We also provided a 30 seconds interval between each block of images, during which only a black screen was displayed, to relax the user and avoid fatigue.

To verify and validate if our procedure had any error and if it was completely clear to the subjects, we performed a pilot test with a 27 years old male and a 18 years old female. With the exception of an image that was duplicated, none of the subjects had any doubt or detected any error in our study. An interesting aspect identified in this pilot test was the different sensitivities of the participants to the negative images. One subject considered the majority of the images very violent, while the other considered them almost neutral, and in some cases he enjoyed the consider negative content.

IV. Emotional Classification Procedure

In this section, we describe the procedure used to classify each image based on the participants' ratings both in terms of the dominant polarity and discrete emotions.

To assign an emotional polarity to an image, we chose the polarity with the highest number of votes. In Table III, we present examples of the distribution of votes across each polarity, while Figure 3 depicts the corresponding images.

TABLE III
Examples of the distribution of votes across each polarity.

Image	Negative	Neutral	Positive	Assigned Polarity
1460.jpg	0.0%	13.3%	86.7%	Positive
Sn087.jpg	20.0%	68.3%	11.7%	Neutral
9925.jpg	40.0%	50.0%	10.0%	Neutral
Sp044.jpg	40.0%	50.0%	10.0%	Neutral
3017.jpg	75.0%	20.0%	5.0%	Negative

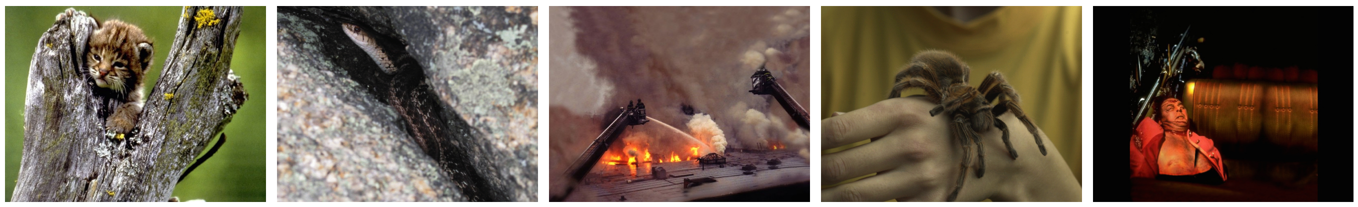
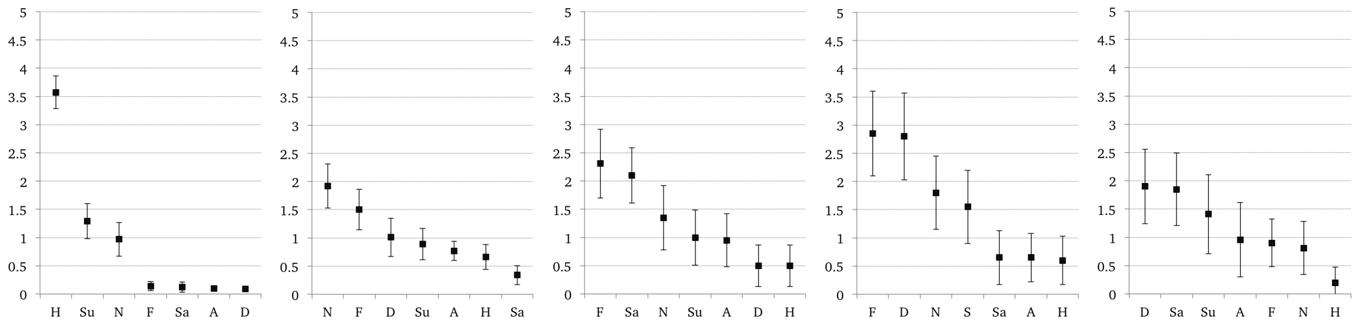

(a) 1460.jpg (b) Sn087.jpg (c) 9925.jpg (d) Sp044.jpg (e) 3017.jpg

Fig. 3. Examples of images from our dataset depicting: (a) kitten, (b) snake, (c) fire, (d) spider, and (e) mutilation.



(a) single (b) blended: 2 emotions (c) blended: 3 emotions (d) blended: 4 emotions (e) undifferentiated

Fig. 4. Examples of Confidence Intervals of images from our dataset, and how they are classified according to our procedure: (a) happiness emotion, (b) neutral and fear emotions, (c) fear, sadness and neutral emotions, (d) fear, disgust, neutral, and sadness emotions, and (e) undifferentiated.

We considered that an image could transmit up to four emotions, with no constraints about their polarity. We made this decision because Posner *et al.* stated that "*individuals do not experience, or recognize, emotions as isolated, discrete entities, but that they rather recognize emotions as ambiguous and overlapping experiences*" [31].

To identify the dominant emotions for each image, we followed the procedure from Mikels *et al.* [16]. However, since we are considering more emotions per image than Mikels (four vs three), our procedure is slightly different. For each image, we computed the mean of the ratings assigned by participants to each emotion, and a 90% t-based Confidence Interval (CI) around each mean. Then, the emotions' label was determined according to the overlap of the CIs for each emotion.

If the mean for one emotion is higher than the means of all the other emotions, and if the CI for that emotion does not overlap with the CIs for the other emotional labels, it is classified as a single emotion (see Figure 4a). If two, three or four means are higher than the rest, and the intersection between their CIs is not empty, the image is categorized as blended (see Figures 4b - 4d). If more than four CIs overlap, the image is classified as undifferentiated (see Figure 4e).

In our study, and contrary to what Mikels did, we could have images with a mix of negative and positive emotions.

V. Results

In this section, we present the polarities agreement and emotional labels assigned to each image. We also present the most elicited emotions together. Finally, we present observations made by our participants during the study.

A. Agreement of Polarity Among Users

In Figures 5 and 6 we can observe, in detail, the votes of the users for each image in our dataset. From the 82 images classified as negative, 77 images had more than 50% of negative votes. The remaining votes were mainly neutral (45 images were rated with at most 30% of neutral votes, while 47 images had at most 5% of positive votes).

Regarding the 66 images classified as neutral, 62 of images had more than 50% of neutral votes. The remaining votes were usually rated more often as negative than positive (37 images with at most 30% of negative votes, while 41 had at most 15% of positive votes). Finally, all the 21 images classified as positive had more than 50% of positive votes. Eighteen images had at most 5% of negative votes, while 10 were rated with at most 30% of neutral votes.

In summary, all polarities were very well identified. When there was some mixing with either the positive or negative polarity, they were mixed with the neutral polarity. For the neutral polarity, it was mainly mixed with the negative polarity.

B. Agreement of Polarity Among Datasets

We compared our results only with GAPED because IAPS does not provide polarity information, and although Mikels provides information about the polarity, it was classified by the authors not by the participants.

We analyzed 76 images (33 negative, 9 positive, and 34 neutral) from the GAPED. For the neutral and positive polarities, we achieved an agreement of 100% for each. For the negative, the achieved agreement was 69%. The biggest mixed was with the neutral polarity (28%), while the mix with the positive polarity was very small (3%).

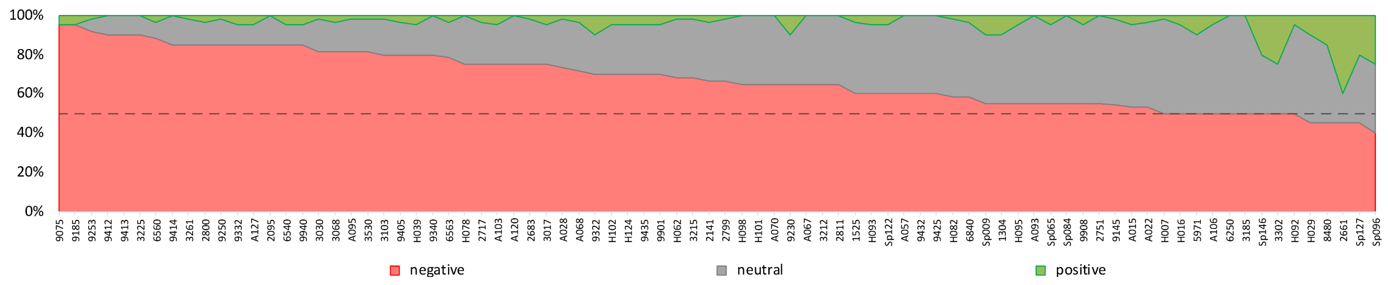


Fig. 5. Images classified as negative in our dataset. We show the percentage of votes that users assigned to each polarity. (best seen in color)

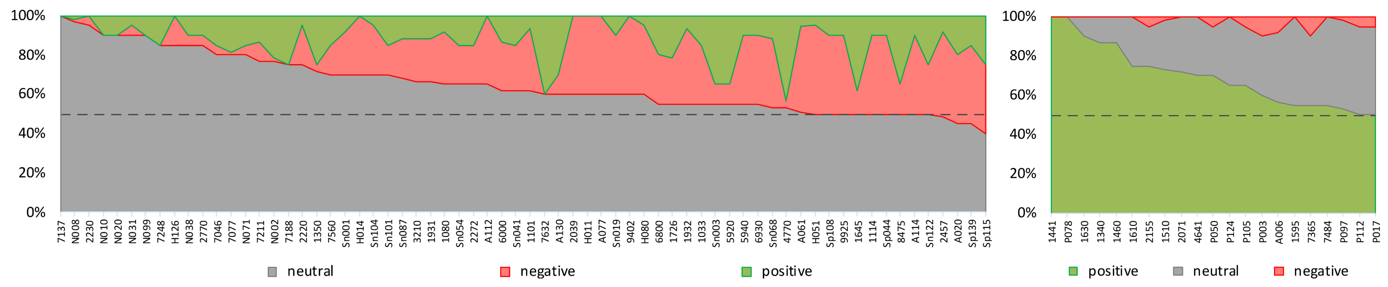


Fig. 6. Images classified as neutral (left) and positive (right) in our dataset. We show the percentage of votes that users assigned to each polarity. (best seen in color).

Dan-Glauser *et al.* also reported that their results in GAPED had a high percentage of negative valence ratings overlapping with the neutral for animal mistreatment, spider, human concern, and snake pictures [13].

C. Valence and Arousal Space

In Figure 7, we present the distribution of the ratings in the valence and arousal space.

For each polarity, a polygon delimits the space in which all pictures of the same polarity are found. If we compare the distribution of the polarities with the emotions displayed in Figure 1, we can see that there is a clear correspondence between the negative emotions with the negative polarity, as well as between the neutral emotion and the neutral polarity.

For the positive polarity, this correspondence with the happiness emotion is not so obvious, but it is easy to see that there is no overlap between the negative and positive polarities. Finally, we can see that there is some confusion between the neutral and negative polarities, as well as between the neutral and positive ones, however less significant.

D. Emotional Labels

From the 169 images of our dataset (see Table IV), we obtained 60 images annotated with a single emotion (35.5%), 87 classified as blended (51.5%), with 29 referring to the combination of two emotions (17.2%), 31 to three emotions (18.4%), and 27 for four emotions (16.0%). Finally, we only had 22 images classified as undifferentiated (13.0%).

If we compare our results with those presented in Mikels dataset (see Table V), we obtained more 6% of images classified with a single emotion, and less 8% undifferentiated images (29.0% (15.98+13.02) vs 36.9% in Mikels considering only three emotions [16]).

If we consider up to four emotions in an image, we have less 24% undifferentiated images (13.0% vs 36.9% in Mikels), while in the case of blended images we have around 21% more images (51.6% vs. 30.5% in Mikels).

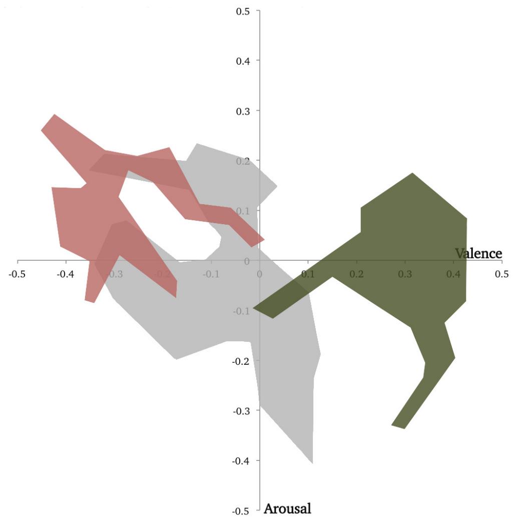


Fig. 7. Representation of the ratings in the valence/arousal space for each polarity. The red area (on the left) corresponds to the negative polarity. The grey area (at the center) corresponds to the neutral polarity, while the green (on the right) corresponds to the positive polarity. (best seen in color).

TABLE IV
DISTRIBUTION OF UL-EPS DATASET CONCERNING THE EMOTIONAL LABELS: SINGLE, BLENDED, AND UNDIFFERENTIATED.

Single	Blended 2	Blended 3	Blended 4	Undifferentiated
60	29	31	27	22
35.50%	17.16%	18.43%	15.98%	13.02%

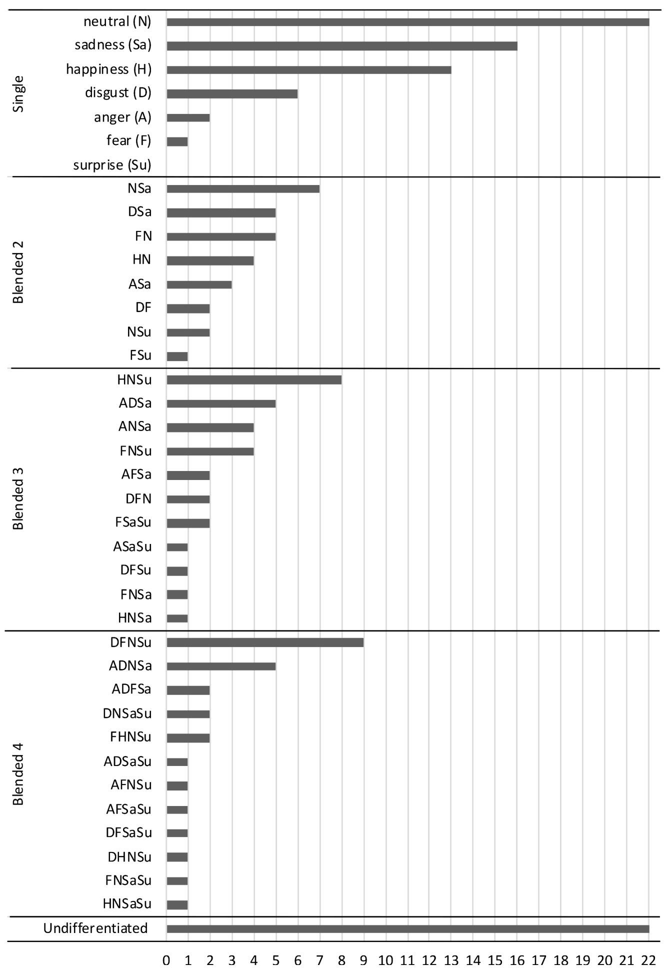


Fig. 8. Emotional labels that result from the classification process.

On the whole, we have a larger number of images annotated with emotional data. Thus, our dataset is more informative about the emotional labels assigned to images.

In Figure 8, we can see the different emotional labels resulting from the classification process. For the blended, we have for example DFSu, i.e., an image that contains the emotions disgust, fear, and surprise. The resulting label does not take into account the weight of each of the emotions present in an image, i.e., label DFSu includes the following combinations: DFSu, DSuF, FDSu, FSuD, SuDF, and SuFD.

E. Relationship Between Emotions

In Figure 9, it is possible to analyze the most elicited single emotions, and the relationship between two emotions. For that, we considered the frequency of occurrence of each emotion.

TABLE V
COMPARISON BETWEEN UL-EPS AND MIKELS DISTRIBUTION OF EMOTIONAL LABELS.

Dataset	Single	Blended (2 and 3)	Undifferentiated
ULEPS	35.50%	35.59%	29.00%
Mikels	30.00%	30.51%	36.92%

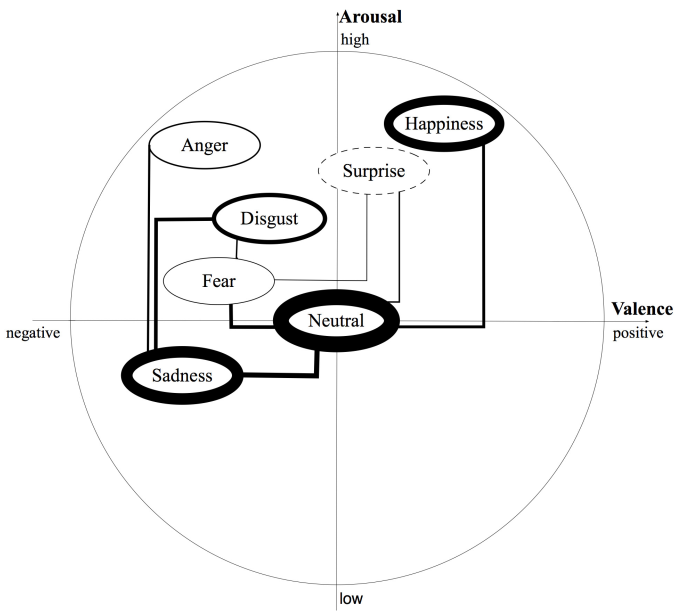


Fig. 9. Relationship between single emotions, and between the blending of two emotions.

The thicker the line, the bigger is the number of images that elicited that emotion (single) or the greater is the relationship between two emotions (blended). A dashed line indicates that there were no images that elicited that emotion.

The most elicited single emotions were neutral, happiness, sadness, and disgust. The most obvious relations are between the emotions neutral and sadness, neutral and fear, neutral and happiness, and sadness and disgust. Anger, fear, and surprise are the emotions less elicited alone. However, surprise tends to appear in conjunction with fear and neutral emotions. In the case of anger, there is some relation with the elicitation of sadness, while fear was elicited together with disgust, as well as with surprise and neutral.

Regarding the correlations between basic emotions, and considering Figures 8 and 9, we confirm the results reported in the literature. Happiness negatively correlates with all the other basic emotions. Anger shows correlation with fear and disgust. There is also correlation between sadness and fear, and between sadness and disgust. Finally, fear was also correlated with disgust. Overall, our results are in line with those reported in previous studies [26], [40]–[42].

F. Observations from Participants

During each session, participants were encouraged to share with us their opinions/comments about the experience. More than 40% of the participants mentioned some type of difficulty in understanding the content of some of the images, leading to confusion about their feelings.

The majority identified the lack of context as the main reason for this, e.g., some participants did not understand if an animal in front of a car will be hit by it or not. In this case there is confusion between feeling negative if the animal is hit, and neutral or positive otherwise.

Five participants claimed that surprise is subjective, difficult to understand, and also difficult to elicit from an image. There seemed to be some exceptions to this, such as a shark moving as it is attacking a person or images with unexpected content like a lamp or stairs. A couple of participants indicated us that none of the images was able to trigger anger.

Regarding the personal taste of the participants, some appreciated specific content such as snakes (4), spiders (3), or aquatic animals (1), while others did not appreciate it at all. However, some of them considered images with those animals "beautiful", mainly due to the colors in them. Three participants declared that they were not sensitive to some images, such as a children smiling, leading them to feel neutral, although they considered that they should feel "happy". Finally, some participants also mentioned that the emotional content of the previous visualized image may interfere in the way they were feeling at that moment.

VI. Conclusion

We described an unrestrained study performed with 60 participants to annotate a dataset of images with the polarity and discrete emotions elicited by each image. During our study there were no restrictions in the selection of the emotions, being possible for a user to associate a positive and a negative emotion to the same image.

We presented the relationship between multiple emotions that arouse when visualizing an image, and we verified that they were in line with existing literature. Moreover, we also presented our experimental procedure designed to avoid stress and fatigue, providing more comfort to the users.

We made a more complete and realistic picture dataset composed of 169 images publicly available to the community[3], as a new contribution to complement the already existing datasets. Each image was annotated with the emotional polarities (positive, neutral, and negative), discrete emotions (anger, disgust, fear, happiness, neutral, sadness, and surprise), and the original valence and arousal information.

Having in mind all the inherent subjectivity of emotions, the different constraints that could affect the participants judgement (current emotional state of the user, user's ability to evaluate what they felt, among others), the overall good agreement among participants, and between our dataset and the GAPED dataset, we can consider that the results achieved by our study are reliable and useful for the elicitation of emotions.

As future work, we intend to use our procedure to annotate more images with polarities, discrete emotions, and the physiological signals collected from the users while viewing the images.

Acknowledgment

This work was supported by national funds through Fundação para a Ciência e Tecnologia, under LASIGE Strategic Project - UID/CEC/00408/2013.

[3]http://www.di.fc.ul.pt/~mjf/research/ul-eps/UL-EPS_2018.xlsx

References

[1] A. R. Damasio, *Descartes' Error: Emotion, Reason, and the Human Brain*. Harper Perennial, 1995.

[2] R. Picard, "Affective computing," MIT Media Laboratory, Perceptual Computing Section, Tech. Rep. 321, 1995.

[3] P. Dunker, S. Nowak, and C. Lanz, "Content-based Mood Classification for Photos and Music," in *Multimedia Information Retrieval*, 2008.

[4] D. Joshi, R. Datta, E. Fedorovskaya, Q.-t. Luong, J. Z. Wang, L. Jia, and J. Luo, "Aesthetics and Emotions in Images [A computational perspective]," *Signal Processing Magazine*, 2011.

[5] D. O. Bos, "EEG-based Emotion Recognition: The Influence of Visual and Auditory Stimuli," *Capita Selecta Paper*, 2006.

[6] L. Axelrod and K. S. Hone, "Affectemes and allaffects: A novel approach to coding user emotional expression during interactive experiences," *Behaviour & Information Technology*, 2006.

[7] M. Tkalčič, A. Kosir, and J. Tasic, "Affective recommender systems: the role of emotions in recommender systems," in *Workshop on Human Decision Making in Recommender Systems*, 2011.

[8] K. C. Klauer, "Affective priming," *European Review of Social Psychology*, 1997.

[9] S. Wang and X. Wang, "Emotion semantics image retrieval: An brief overview," in *Affective Computing and Intelligent Interaction*, 2005.

[10] T. Li and M. Ogihara, "Detecting emotion in music," in *Music Information Retrieval*, 2003.

[11] K. Sun, J. Yu, Y. Huang, and X. Hu, "An improved valence-arousal emotion space for video affective content representation and recognition." Multimedia and Expo, 2009.

[12] V. L. Rubin, J. M. Stanton, and E. D. Liddy, "Discerning Emotions in Texts," in *Exploring Attitude and Affect in Text: Theories and Applications*, 2004.

[13] E. S. Dan-Glauser and K. R. Scherer, "The Geneva affective picture database (GAPED): A new 730-picture database focusing on valence and normative significance." *Behavior Research Methods*, 2011.

[14] P. Lang, M. Bradley, and B. Cuthbert, "International affective picture system (IAPS): Affective ratings of pictures and instruction manual." NIMH Center for the Study of Emotion and Attention, Tech. Rep., 1997.

[15] A. Marchewka, Ł. Zurawski, K. Jednoróg, and A. Grabowska, "The Nencki Affective Picture System (NAPS): introduction to a novel, standardized, wide-range, high-quality, realistic picture database." *Behavior Research Methods*, 2014.

[16] J. A. Mikels, B. L. Fredrickson, G. R. Larkin, C. M. Lindberg, S. J. Maglio, and P. A. Reuter-Lorenz, "Emotional category data on images from the International Affective Picture System." *Behavior Research Methods*, 2005.

[17] A. Smith, "A new set of norms." *Behavior Research Methods*, 2004.

[18] ——, "Smith2004norms.txt," *Retrieved October 2, 2004 from Psychonomic Society Web Archieve*, 2004.

[19] M. Riegel, Ł. Zurawski, M. Wierzba, A. Moslehi, Ł. Klocek, M. Horvat, A. Grabowska, J. Michałowski, K. Jednoróg, and A. Marchewka, "Characterization of the Nencki Affective Picture System by discrete emotional categories (NAPS BE)." *Behavior Research Methods*, 2016.

[20] E. Fox, *Emotion Science: Cognitive and Neuroscientific Approaches to Understanding Human Emotions*. Palgrave Macmillan, 2008.

[21] R. D. Lane, E. M. Reiman, G. L. Ahern, G. E. Schwartz, and R. J. Davidson, "Neuroanatomical correlates of happiness, sadness, and disgust." *The American journal of psychiatry*, 1997.

[22] Y. Choi and E. M. Rasmussen, "Searching for images: The analysis of users' queries for image retrieval in American history," *Journal of the American Society for Information Science and Technology*, 2003.

[23] K. A. Olkiewicz and U. Markowska-kaczmar, "Emotion-based image retrieval - An artificial neural network approach," in *Computer Science and Information Technology*, 2010.

[24] A. Hanjalic, "Extracting Moods from Pictures and Sounds: Towards Truly Personalized TV," *Signal Processing Magazine*, 2006.

[25] C. E. Izard, *The Psychology of Emotions*. Plenum Press, 1991.

[26] S. Schmidt and W. Stock, "Collective indexing of emotions in images. A study in emotional information retrieval," *Journal of the American Society for Information Science and Technology*, 2009.

[27] R. Plutchik, "The Nature of Emotions," *American Scientist*, 2001.

[28] P. Ekman and R. J. Davidson, *The nature of emotion : fundamental questions*. New York : Oxford University Press, 1994.

[29] O. da Pos and P. Green-Armytage, "Facial expressions, colours and basic emotions," *Journal of the International Colour Association*, 2007.

[30] Y. Liu, O. Sourina, and M. K. Nguyen, "Real-Time EEG-Based Emotion Recognition and Its Applications," *Transactions on Computational Science*, 2011.

[31] J. Posner, J. a. Russell, and B. S. Peterson, "The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology." *Development and psychopathology*, 2005.

[32] J. Yoon, "Utilizing quantitative users' reactions to represent affective meanings of an image," *Journal of the American Society for Information Science and Technology*, 2010.

[33] J. Machajdik and A. Hanbury, "Affective image classification using features inspired by psychology and art theory," *International Conference on Multimedia*, 2010.

[34] M. Wessa, P. Kanshe, P. Neumeister, K. Bode, J. Heissler, and S. Schoenfelder, "EmoPics: Subjektive und psychophysiologische Evaluationen neuen Bildmaterials für die klinisch-bio-psychologische Forschung," *Zeitschrift für Klinischer Psychologie und Psychotherapie, Supplement, 1/11, 77*, 2010.

[35] D. Lundqvist, A. Flykt, and A. Öhman, *The Karolinska Directed Emotional Faces - KDEF.* CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, 1998.

[36] N. Tottenham, A. Borscheid, K. Elertsen, D. Marcus, and C. Nelson, "Categorization of facial expressions in children and adults: Establishing a larger stimulus set." *Cognitive Neuroscience Society*, 2002.

[37] M. M. Bradley and P. J. Lang, "Measuring emotion: The self-assessment manikin and the semantic differential," *Journal of Behavior Therapy and Experimental Psychiatry*, 1994.

[38] M. Tkalčič, U. Burnik, and A. Košir, "Using affective parameters in a content-based recommender system for images," *User Modeling and User-Adapted Interaction*, 2010.

[39] P. Arriaga and G. Almeida, "Fábrica de emoções : A eficácia da exposição a excertos de filmes na indução de emoções," Instituto Superior de Psicologia Aplicada, Tech. Rep., 2010.

[40] I. Bretherton and M. Beeghly, "Talking about internal states: The acquisition of an explicit theory of mind," *Developmental Psychology*, 1982.

[41] B. Fehr and J. Russell, "Concept of emotion viewed from a prototype perspective," *Journal of Experimental Psychology: General*, 1984.

[42] P. Shaver, J. Schwartz, D. Kirson, and C. O'Connor, "Emotion knowledge: Further exploration of a prototype approach." *Journal of Personality and Social Psychology*, 1987.

Event-Based Data Input, Modeling and Analysis for Meditation Tracking using TDR System

Shi-Kuo Chang[1], CuiLing Chen[2], Wei Guo[1] and NanNan Wen[1]
[1]School of Computing and Information
University of Pittsburgh, Pittsburgh, PA 15238, USA
{schang, weg21, naw66}@pitt.edu

[2]College of Mathematics and Statistics, Guangxi Normal University, Guilin 541004, PR China
mathchen@163.com

Abstract

In this paper we describe an experimental TDR system with continuous data input from devices such as smart phones and sensors such as brain wave headsets. We developed event-based data input, modeling and analysis techniques in order to analyze input data and track progress of meditation. Initial experimental results indicate that this approach is quite promising.

Keywords

Meditation tracking, event-based data input, modeling and analysis, slow intelligence system, TDR system.

1 Introduction

In our previous work we developed the TDR system, which is a multi-level slow intelligence system with interacting super-components each of which has its own computation cycle [1], as a platform to explore applications in personal health care, emergency management, social networks and so on. In this paper we apply the TDR system to event-based data analysis and visualization for meditation tracking.

Meditation, defined as “the attention inwards towards the subtler levels of a thought until the mind transcends the experience of the subtlest state of the thought and arrives at the source of the thought”, has been proven to have positive effects on social skills, feeling of compassion, self-management, somatic awareness and mental flexibility. It has also been used in treatment of anxiety disorders, stress reduction, chronic pain, persistent pain, depression, autism spectrum disorders, traumatic experiences, acquired brain injury, and even eating disorder, psoriasis and substance abuse.

Nowadays many people are learning meditation. However there are still no adequate meditation monitoring systems to take continuous measurements from various sensors when a person is in meditation and to track its progress. In this paper we describe an experimental TDR system with continuous data input from devices such as smart phones and sensors such as brain wave headsets. We developed event-based data input, modeling and analysis techniques in order to analyze input data and track progress of meditation. Initial experimental results demonstrate that this approach is quite promising.

The paper is organized as follows. In Section 2 we describe the system architecture. The interface to support event-based data input is presented in Section 3. Event-based data modeling is described in Section 4, followed by a detailed example of data analysis presented in Section 5. Section 6 presents user scenarios for the experimental system. Discussion and conclusion are presented in Section 7.

2 System Architecture

Figure 1 illustrates the experimental TDR system consisting of interacting super-components and the chronobot database. Each super-component has its own computational cycles. The super-components interact with one another through the SIS server. Based on requests from the administrator, the super-components process input data and upload them to the Chronobot database. In the TDR system there are at least three super-components: Tien (Heaven), Di (Earth) and Ren (Human). The Tien super-component handles sensors for the atmosphere, the Di super-component deals with sensors for the lithosphere and the Ren super-component manages sensors for the human body.

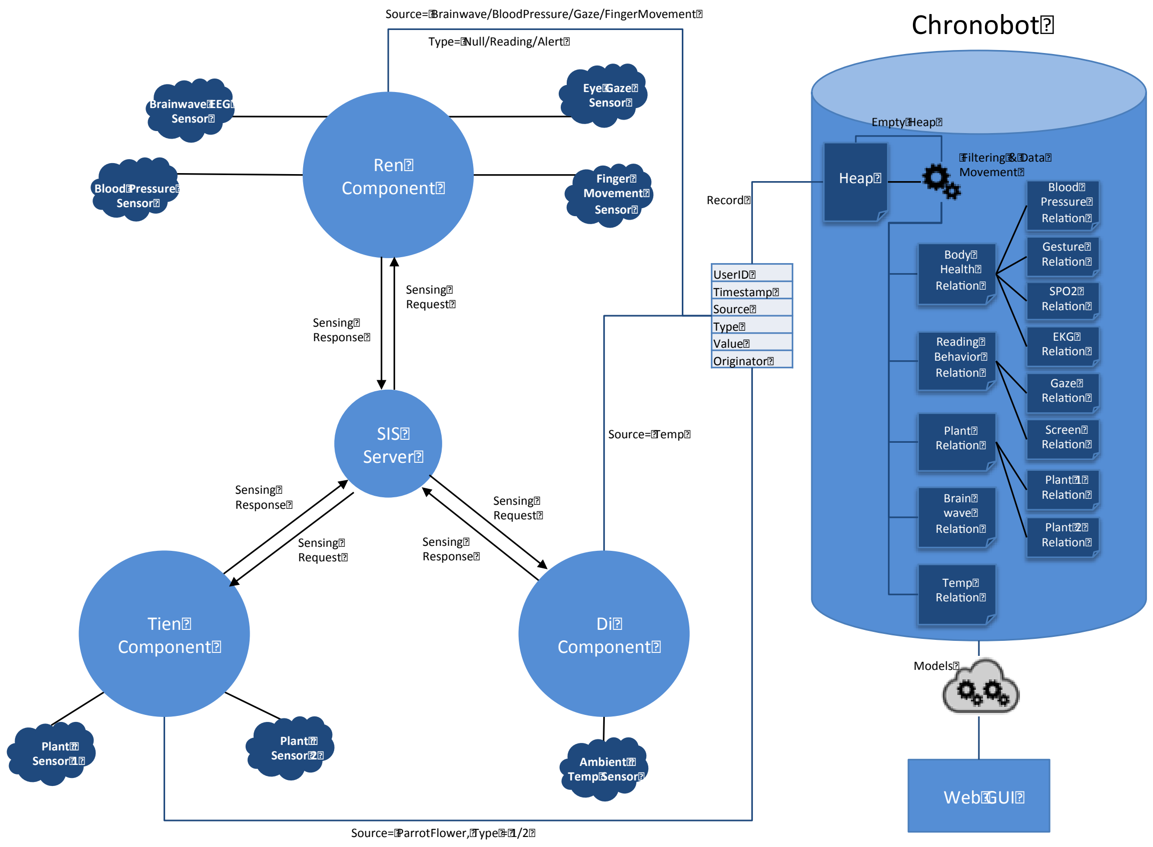


Figure 1. TDR System with super-components and Chronobot database.

In order to track meditation we proposed to use brain wave headset as well as eye gaze tracking by smart phone during meditation [2]. Data from brain wave sensor and eye gaze tracker are collected by their respective input processors in Ren super-component and uploaded to the Heap relation in the Chronobot database. The Heap is a collection of records each with a variable number of attributes for different types of sensor data, which are filtered and moved into different relations such as Gazing Behavior Relation, Brain Wave Relation and so on, by the request of the administrator through the Web GUI.

A more detailed view is shown in Figure 2. Records in the Heap are first filtered and then moved to the corresponding relations. In the filtering of data, the resultant data must conform to the model for the corresponding relation. We will first explain the conceptual framework. The detailed formal model will be presented in Section 4.

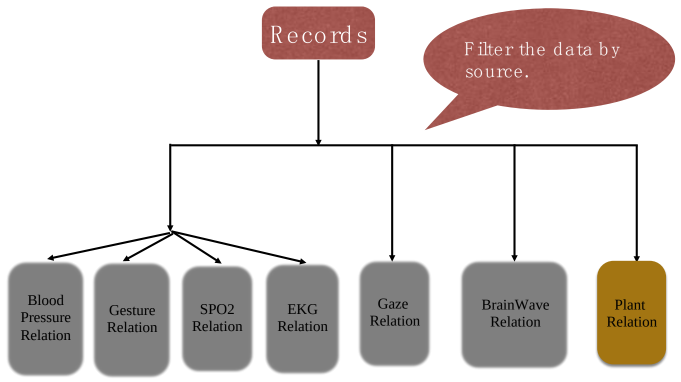


Figure 2. Records are filtered and moved to the corresponding relations.

3. Event-Based Data Input

The database for the TDR system is a time varying database. To make sense of the time varying database, we need to monitor the data streams and detect significant changes. For the best of our knowledge, there's few researches on designing user interface for time varying database. User interface design requires a good understanding of user needs [3], in our approach we need to be able to specify what is normal and what is not normal. In fact, a database is governed by a data model specifying what is the normal pattern. The computation cycles specify the collection, filtering and storage of data that conform to the normal pattern, so the end result is a **normal event**. The cycle then repeats itself. When the data deviates from the normal pattern, it is an **abnormal event** to take notice of. Our approach to user interface design is therefore based upon this concept of normal and abnormal events.

In recent years, visualization has become an important tool to support exploration and analysis of large volumes of data. Therefore, to shift the needs of users into the focus, we should pursue an event-based approach to visualization. This approach allows users to specify their interests as event types. The **normal event** is the data model. The **abnormal event** is what deviates from the data model.

During a computation cycle, the normal event is usually the end result, i.e. the processing and storage of data that conforms to the specified data model. When instances of the specified abnormal event types are detected, the user interface automatically adjusts visual representations according to the detected event instances. This approach results in visualizations that are adapted to the needs and interests of the users. Hence, users are supported in achieving their task at hand.

In terms of event-based visualization, the basic idea is to let users specify their interest by means of event types, to detect instances of these events in the data, and to create representations that can be automatically adjusted with respect to the detected event instances. Accordingly, three main aspects are investigated:.

1. Event specification,
2. Event detection,
3. Event representation.

To bridge the gap between informal user interests and the digital language of computers, a formalism for the event specification must be developed. Here, the difficulty is to build a formal basis that provides a suitable expressiveness while still allowing users to specify their interests as easily as possible. Especially when facing users who are not familiar with event-based visualization, it is essential to provide methods and tools that allow an intuitive specification of event types.

The task of the event specification is to compile event types that are or might be of interest to visualization users. The event specification necessitates a formal foundation to allow a later detection of event instances.

In our approach, we have two types of events: normal events that represent the data model, and abnormal events that represent deviation from the data model. Events are always specified for a certain relation. Before moving data from heap to relation, we first check if the tuple satisfies a certain event type.

(1) **Normal Event:** As an example, if every tuple has an error rate less than the threshold ε (for example $\varepsilon = 0.1$), then it is a normal event. This event can be described as:

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) \le \varepsilon, \qquad \text{..........................} \ (1)$$

Once a normal event is specified, we can create a computation cycle to get the TDR system started. For formal definition, see Section 4.

(2) **Abnormal Event 1:** As an example, for three consecutive tuples, if each tuple's error rate exceeds the threshold ε, then it is an abnormal event. This event can be described as:

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon,$$
$$\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > \varepsilon, \quad \text{......... ....} \ (2)$$
$$\Psi(\Delta T(\eta), X(v_{I+2}), Y(v_{I+2})) > \varepsilon,$$

For formal definition, see section 4. If condition (2) is met, user then can use the following steps to specify this event.:

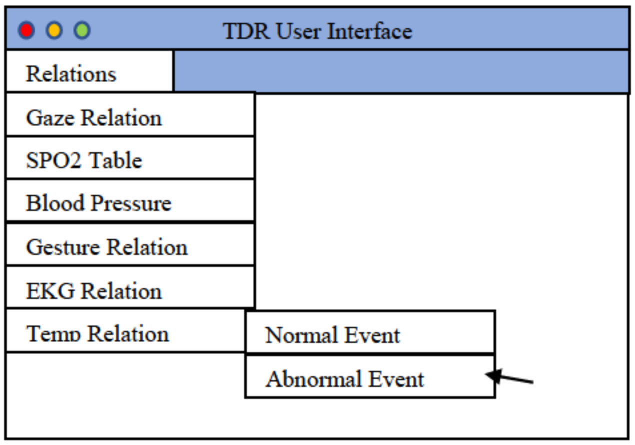


Figure 3. Choose Event Type for a certain relation.

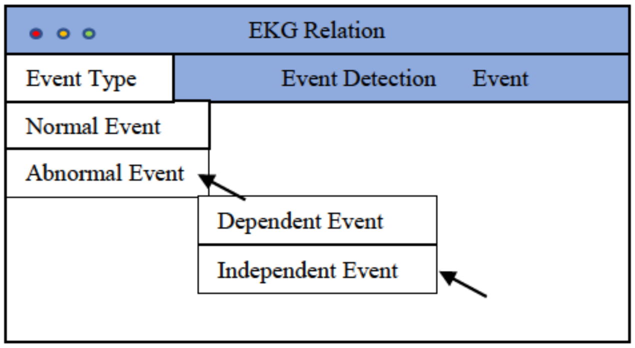


Figure 4. Click on Abnormal Event and then Independent Event.

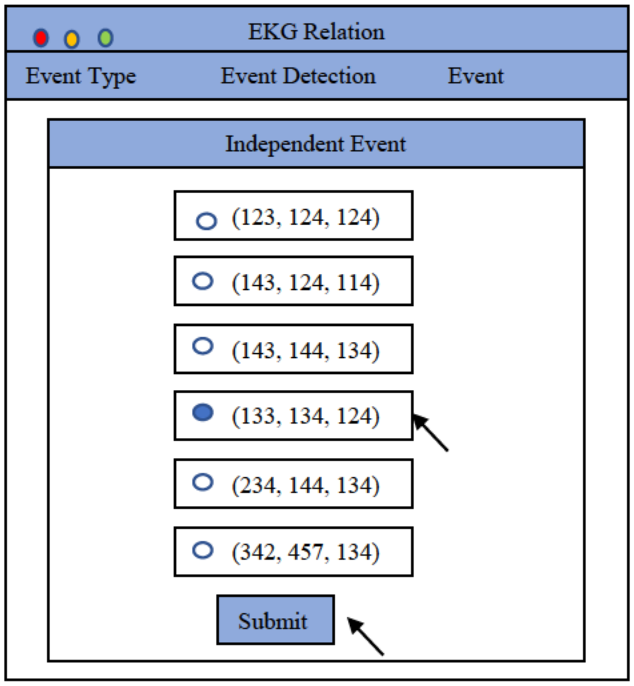


Figure 5. Choose tuple and submit event.

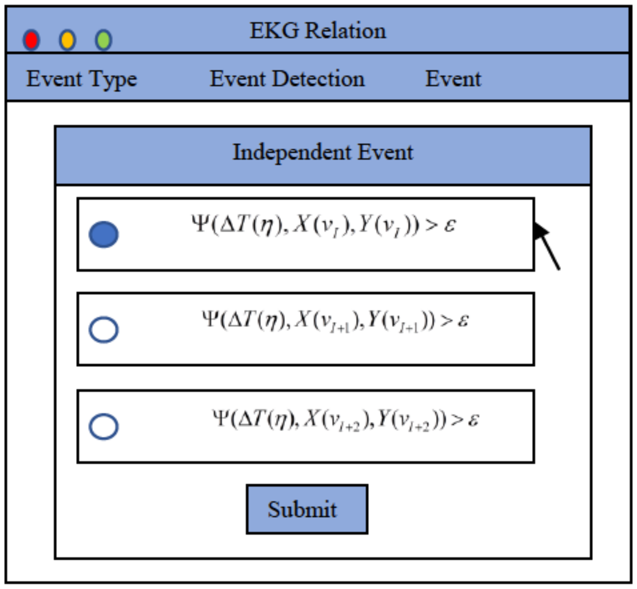

EKG Relation
Event Type
Event Detection
Event
Independent Event
● $\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon$
○ $\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > \varepsilon$
○ $\Psi(\Delta T(\eta), X(v_{I+2}), Y(v_{I+2})) > \varepsilon$
Submit

Figure 6. Choose event type.

(3) **Abnormal Event 2:** For three consecutive tuples, if each tuple's error rate is twice as much as the previous tuple, then it is an event. This event can be described as:

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon \quad \text{..........................} \ (3)$$
$$\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > 2 \cdot \Psi(\Delta T(\eta), X(v_I), Y(v_I))$$
$$\Psi(\Delta T(\eta), X(v_{I+2}), Y(v_{I+2})) > 2 \cdot \Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1}))$$

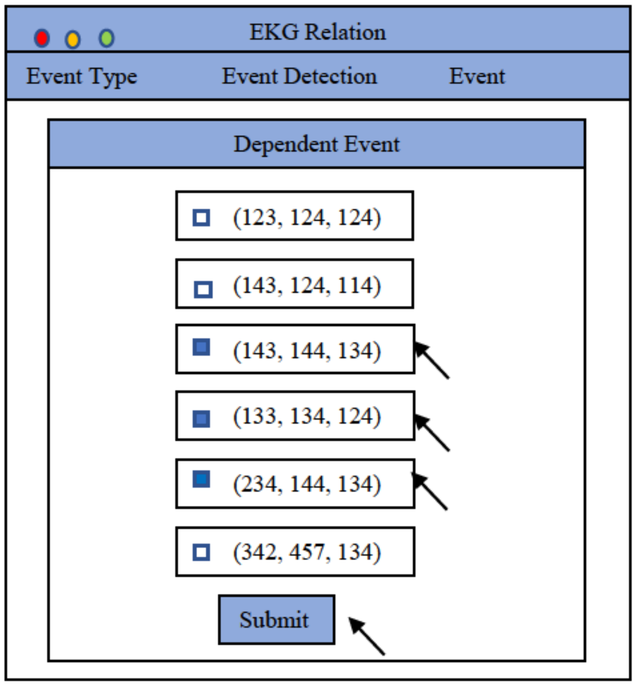


Figure 7. Choose 3 tuples and submit.

The user can click on Abnormal Event and then Dependent Event, similar to Figure 4. For formal definition, see section 4. If condition (3) is met, user then can use the steps in Figure 7 and Figure 8 to specify this event.

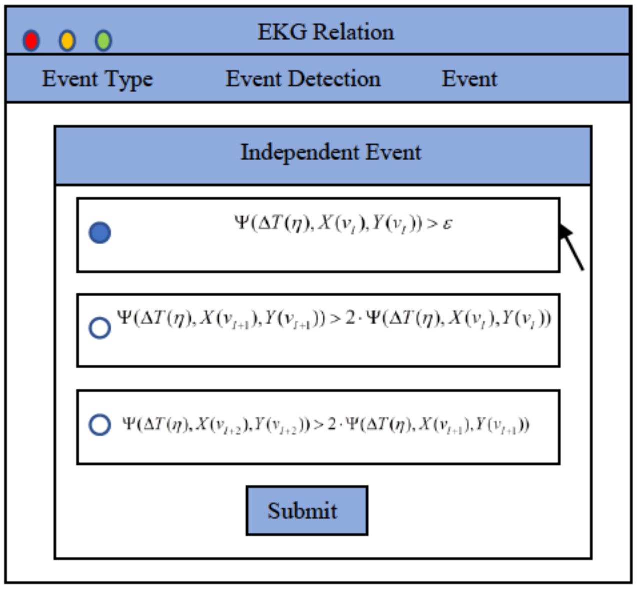

EKG Relation
Event Type
Event Detection
Event
Independent Event
$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon$
$\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > 2 \cdot \Psi(\Delta T(\eta), X(v_I), Y(v_I))$
$\Psi(\Delta T(\eta), X(v_{I+2}), Y(v_{I+2})) > 2 \cdot \Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1}))$
Submit

Figure 8. Choose event type.

4. Event-Based Data Modeling

Inspired by [4] and [5], we consider a multimedia database with time-varying $R(T, A_1, \cdots, A_n)$, where T denotes time, $A_1, \cdots, A_n$ are other attributes, v_i is the tuple corresponding to the moment t_i, U is a set of the attributes $A_1, \cdots, A_n$, and $dom(A_k)$ is the domain of each A_k, $v_i[A_k]$ denotes the value of the tuple v_i in the attribute A_k. Thus for any two moments t_i and t_j of T in R, there are always a pair of corresponding tuples $v_i = (t_i, v_i[A_1], \cdots, v_i[A_n])$, $v_j = (t_j, v_j[A_1], \cdots, v_j[A_n])$.

The similarity between any two attribute values $v_i[A_k]$ and $v_j[A_k]$ of A_k is based on a distance function of type $d: dom(A_k)^2 \to [0,1]$ [4]. For simplicity, we denote with $D(A_k)$ the set of the distance functions defined on A_k.

According to the distance function of type $d: dom(A_k)^2 \to [0,1]$, definition 1 is given as follows, then from which we get definition 2.

Definition 1 Given a relation $R(T, A_1, \cdots, A_n)$, for a pair of tuples v_i and v_j corresponding to any two moments t_i and t_j of T, we say that v_i is **similar** within τ to v_j with respect to d at the moments t_i and t_j, denoted with $\boldsymbol{v_i[A_k] \cong_{(d,\tau,t_i,t_j)} v_j[A_k]}$, iff $\boldsymbol{d(v_i[A_k], v_j[A_k]) \le \tau}$, where τ is a threshold.

Definition 2 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, we say the **type-M function dependency** during the time of $\boldsymbol{T}$ (**T-MFD**): $\boldsymbol{X_{(d_1,\tau')} \xrightarrow{T} Y_{(d_2,\tau'')}}$ holds, if and only if for a pair of tuples v_i and v_j corresponding to any two moments t_i and t_j of T, whenever $v_i[X] \cong_{(d_1,\tau',t_i,t_j)} v_j[X]$, then $v_i[Y] \cong_{(d_2,\tau',t_i,t_j)} v_j[Y]$, where $d_1 \in D[X]$, $d_2 \in D[Y]$, $\tau', \tau'' \in [0,1]$ are thresholds.

Obviously, given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, sometimes there are a pair of tuples v_I and v_J corresponding to some two moments t_I and t_J of T such that $v_I[X] \cong_{(d_1,\tau',t_I,t_J)} v_J[X]$ holds, whereas $v_I[Y] \cong_{(d_2,\tau'',t_I,t_J)} v_J[Y]$ doesn't hold. Then the following definition is necessary.

Definition 3 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, if $v_I[X] \cong_{(d_1,\tau',t_I,t_J)} v_J[X]$ holds, whereas $v_I[Y] \cong_{(d_2,\tau'',t_I,t_J)} v_J[Y]$ doesn't hold, we say v_I, v_J at the moments t_I and t_J with respect to X, Y constitute a **dependency violation event** (**DVE**) of $\boldsymbol{T}$, denote by **T-DVE** $\boldsymbol{-v_I, v_J[X, Y]}$. The DVEs of all tuples during the time of T with respect to X, Y are denoted as **T-DVEs** $\boldsymbol{-X, Y}$.

Thus the occurrence rate of T-DVEs $-X, Y$ with respect to any two attributes X, Y is a very important problem that one concerns, which can be calculated by the following definition.

Definition 4 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, we define the **dependency violation**

rate (**DVR**) of X,Y during the time of $\boldsymbol{T}$ (**T-DVR -*X,Y***) as follows:

$$\Psi(T,X,Y)=\frac{r_1}{r_T},$$

where r_T , $r_1 \subseteq r_T$ denote the combinatorial number of any pair of attributes in X or Y during the time of T , the number of the T-DVEs -X,Y , respectively.

We know if there are T-DVEs-X,Y , and the T-DVR-X,Y is very small, even very close to zero, then $X_{(d_1,\tau')} \xrightarrow{T} Y_{(d_2,\tau'')}$ almost holds, from which the following generalized T-MFD definition is yielded.

Definition 5 Given a relation $R(T,A_1,\cdots,A_n)$ and $X,Y \subseteq U$, we say the **relaxed type-M function dependency** during the time of $\boldsymbol{T}$ (**T-RMFD**): $\boldsymbol{X}_{(d_1,\tau)} \xrightarrow{\Psi(T,X,Y)\le\varepsilon} \boldsymbol{Y}_{(d_2,\tau)}$ holds, if and only if for a pair of tuples v_i and v_j corresponding to any two moments t_i and t_j of T , whenever $v_i[X] \cong_{(d_1,\tau',t_i,t_j)} v_j[X]$, then almost $v_i[Y] \cong_{(d_2,\tau'',t_i,t_j)} v_j[Y]$ holds, and $\Psi(T,X,Y)\le\varepsilon$, where $\Psi(T,X,Y)$ is the T-DVR -X,Y , $d_1 \in D[X]$, $d_2 \in D[Y]$, and $\tau',\tau'',\varepsilon \in [0,1]$ are thresholds.

Remark 1: It is depended on the value of ε to a great degree whether $X_{(d_1,\tau')} \xrightarrow{\Psi(T,X,Y)\le\varepsilon} Y_{(d_2,\tau'')}$ holds.

For a relation $R(T,A_1,\cdots,A_n)$ and $X,Y \subseteq U$, when T is too long and there are too many data during the whole time T , we can consider to investigate fewer data during a part time. If we use the symbol $\boldsymbol{\eta}$ to denote the duration of the part time, and get the following definition.

Definition 6 Given a relation $R(T,A_1,\cdots,A_n)$ and $X,Y \subseteq U$, we say the **relaxed type-M function dependency** during ***ΔT(η)*** (***ΔT(η)* - RMFD**) :

$$\boldsymbol{X}_{(d_1,\tau')} \xrightarrow{\Psi(\Delta T(\eta),X,Y)\le\varepsilon} \boldsymbol{Y}_{(d_2,\tau')}$$

holds, if and only if for a pair of tuples v_i and v_j corresponding to any two moments t_i and t_j during $\Delta T(\eta)$ (i.e., $\Delta T = \max_{M_1\le i,j\le M_2, i\ne j} |t_i - t_j| \le \eta$, $M_1, M_2 \in \{1,2,\cdots,m\}$), whenever $v_i[X] \cong_{(d_1,\tau',t_i,t_j)} v_j[X]$ holds, there is almost $v_i[Y] \cong_{(d_2,\tau',t_i,t_j)} v_j[Y]$ holds, and

$$\Psi(\Delta T(\eta),X,Y)=\frac{r_2}{r_{\Delta T(\eta)}}\le\varepsilon,$$

where $r_{\Delta T(\eta)}$ is the combinatorial number of any pair of attributes in X or Y during $\Delta T(\eta)$, $r_2 \subseteq r_{\Delta T(\eta)}$ is the number of the DVEs$-X,Y$ during $\Delta T(\eta)$ (***ΔT(η)* - DVEs -*X,Y***), m is the number of the tuples during the whole time T , $d_1 \in D[X]$, $d_2 \in D[Y]$, and $\tau',\tau'',\varepsilon \in [0,1]$ are thresholds.

Remark 2: Similar to definition 4, we can say $\Psi(\Delta T(\eta),X,Y)$ in definition 6 is the dependency violation rate of X,Y during $\Delta T(\eta)$ (***ΔT(η)* - DVR -*X,Y***) .

For a relation $R(T,A_1,\cdots,A_n)$ and $X,Y \subseteq U$, if we investigate the data during some part time of T and can get the relation between X and Y during the whole time T , then we only need to consider the relation $R(T,A_1,\cdots,A_n)$ during this part time. The following definition describes this case.

Definition 7 Given a relation $R(T,A_1,\cdots,A_n)$ and $X,Y \subseteq U$, if

$$T=\Delta T_1(\eta)\cup\cdots\cup\Delta T_L(\eta),\ \ \Delta T_i(\eta)\cap\Delta T_j(\eta)=\Phi$$
$$(i,j=1,2,\cdots,L,\ \ i\ne j),$$

where $\Delta T_i(\eta)$ denotes $\max_{j\ne k} |t_j - t_k| \le \eta$ ($j,k \in \{1,2,\cdots,L\}$), and $\Psi(\Delta T_i(\eta),X,Y)\le\varepsilon$ $(i=1,2,\cdots,L)$ holds, we use $\boldsymbol{\Psi(\Delta T_I(\eta),X,Y)}$ to denote $\min_{1\le i\le L}\{\Psi(\Delta T_i(\eta),X,Y)\}$. Thus the **RMFD** of X , Y during $\boldsymbol{\Delta T_I(\eta)}$ can be expressed as

$$\boldsymbol{X}_{(d_1,\tau')} \xrightarrow{\Psi(\Delta T_I(\eta),X,Y)\le\varepsilon} \boldsymbol{Y}_{(d_2,\tau'')}.$$

Remark 3: Under the conditions of definition 7, we can get $X_{(d_1,\tau')} \xrightarrow{\Psi(T,X,Y)\le\varepsilon} Y_{(d_2,\tau'')}$ by $X_{(d_1,\tau')} \xrightarrow{\Psi(\Delta T_I(\eta),X,Y)\le\varepsilon} Y_{(d_2,\tau'')}$.

Based on definition 3 and definition 6, it is easy to know there are two classes of dependency violation events during $\Delta T(\eta)$, so we summarize as follows.

Definition 8 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, for a pair of tuples v_I and v_J corresponding to some two moments t_I and t_J during $\Delta T(\eta)$, if there is one of the following cases happening, we say v_I, v_J at the moments t_I and t_J with respect to X, Y constitute a **dependency violation event** during ***ΔT(η)*** (***ΔT(η)*** **-DVE**):

(1) $|t_I - t_J| > \eta$;

(2) $v_I[X] \cong_{(d_1, \tau', t_i, t_j)} v_J[X]$ holds, whereas $v_I[Y] \cong_{(d_2, \tau', t_i, t_j)} v_J[Y]$ doesn't hold.

For simplicity, Case (1) is denoted as ***ΔT(η)*** **- DVE -t_I ,t_J** , Case (2) is denoted as ***ΔT(η)*** **- DVE -v_I ,v_J [X, Y]**.

For a tuple v_I corresponding to some moment t_I during $\Delta T(\eta)$, sometimes we need to know the **DVR** of v_I. To this end we need to introduce the definition of **DVE** of v_I. According to definition 3, we present the following definition.

Definition 9 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, for some tuple v_I corresponding to some moment t_I and a series of tuples v_j corresponding to some moments t_j during $\Delta T(\eta)$, if $v_I[X] \cong_{(d_1, \tau', t_I, t_j)} v_j[X]$ holds, whereas $v_I[Y] \cong_{(d_2, \tau'', t_I, t_j)} v_j[Y]$ doesn't hold $(1 \le I, j \le m_1,\ I \ne j)$, where m_1 is the number of the tuples during $\Delta T(\eta)$, we say v_I, v_j at the moments t_I and t_j with respect to *X,Y* constitute a **dependency violation event** during ***ΔT(η)***, denote by ***ΔT(η)*** **- DVE -v_I , v_j [X, Y]**. All of the DVEs of v_I with respect to *X,Y* during $\Delta T(\eta)$ are denoted as ***ΔT(η)*** **- DVEs -v_I [X, Y]**.

Based on definition 9, we can get definition 10.

Definition 10 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, for the tuple v_I corresponding to some moment t_I during $\Delta T(\eta)$, we define the **dependency violation rate** of v_I during ***ΔT(η)*** (***ΔT(η)*** **- *DVR* - v_I [X, Y]**) as follows:

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) = \frac{r_{v_I}}{m_1 - 1},$$

where $r_{v_I} \subseteq r_{\Delta T(\eta)}$ is the number of $\Delta T(\eta)$-DVEs -v_I[X, Y], m_1 is the number of the tuples during $\Delta T(\eta)$.

For a tuple v_I during $\Delta T(\eta)$ and a given ε, if $\Psi(\Delta T(\eta), X(v_I), Y(v_I)) \le \varepsilon$, then we say v_I constitutes a **normal event** (**NE**) (see section 3). Otherwise we say it is an **abnormal event** (**ANE**). In particular, we study the following cases.

Definition 11 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, for the three consecutive tuples $v_I, v_{I\ 1}, v_{I\ 2}$ corresponding to some moments $t_I, t_{I\ 1}, t_{I\ 2}$ during $\Delta T(\eta)$, if

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon,$$

$$\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > \varepsilon,$$

$$\Psi(\Delta T(\eta), X(v_{I+2}), Y(v_{I+2})) > \varepsilon,$$

we say the tuples v_I, v_{I+1}, v_{I+2} constitute an **abnormal event 1** during ***ΔT(η)*** **(*ΔT(η)* - ANE -1)** (see section 3) .

Definition 12 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, for the three consecutive tuples v_I, v_{I+1}, v_{I+2} corresponding to some moments t_I, t_{I+1}, t_{I+2} during $\Delta T(\eta)$, if

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon,$$

$$\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > 2 \cdot \Psi(\Delta T(\eta), X(v_I), Y(v_I)),$$

$$\Psi(\Delta T(\eta), X(v_{I\ 2}), Y(v_{I\ 2})) > 2 \cdot \Psi(\Delta T(\eta), X(v_{I\ 1}), Y(v_{I\ 1})),$$

we say the tuples v_I, v_{I+1}, v_{I+2} constitute an **abnormal event 2** during ***ΔT(η)*** **(*ΔT(η)* - ANE-2)** (see section 3) .

More generally, we have the following case.

Definition 13 Given a relation $R(T, A_1, \cdots, A_n)$ and $X, Y \subseteq U$, for the three consecutive tuples v_I, v_{I+1}, v_{I+2} corresponding to some moments t_I, t_{I+1}, t_{I+2} during $\Delta T(\eta)$, if

$$\Psi(\Delta T(\eta), X(v_I), Y(v_I)) > \varepsilon,$$

$$\Psi(\Delta T(\eta), X(v_{I+1}), Y(v_{I+1})) > n \cdot \Psi(\Delta T(\eta), X(v_I), Y(v_I)),$$

$$\Psi(\Delta T(\eta), X(v_{I\ 2}), Y(v_{I\ 2})) > n \cdot \Psi(\Delta T(\eta), X(v_{I\ 1}), Y(v_{I\ 1})),$$

where $n > 2$, we say the tuples v_I, v_{I+1}, v_{I+2} constitute an **abnormal event N** during ***ΔT(η)*** (***ΔT(η)*** **- ANE-N**).

Sometimes X and Y don't satisfy definition 6 during $\Delta T(\eta)$ because there are abnormal events. However, the subsets X_I, Y_I of X and Y getting by deleting the abnormal events, maybe satisfy definition 6. The following definition describes this case.

Definition 14 Given a relation $R(T, A_1, \cdots, A_n)$ and

$$X = \left(v_1[X], \cdots, v_n[X]\right) \subseteq U,$$
$$Y = \left(v_1[Y], \cdots, v_n[Y]\right) \subseteq U,$$

if there is v_k corresponding to some moment t_k during $\Delta T(\eta)$ such that

$$\Psi(\Delta T(\eta), X(v_k), Y(v_k)) > \varepsilon$$
$$(k = K_1, K_2, \cdots, K_M \in \{1, 2, \cdots, m_1\}),$$

whereas for

$$X_I = X - \{v_k[X] \mid k \in \{K_1, K_2, \cdots, K_M\}\},$$
$$Y_I = Y - \{v_k[Y] \mid k \in \{K_1, K_2, \cdots, K_M\}\},$$

and any $v_i[X],\ v_j[X] \in X_I$, $v_i[Y],\ v_j[Y] \in Y_I$, whenever $v_i[X] \cong_{(d_1, \tau', t_i, t_j)} v_j[X]$ holds, there is almost $v_i[Y] \cong_{(d_2, \tau', t_i, t_j)} v_j[Y]$ holds, and

$$\Psi(\Delta T(\eta), X_I, Y_I) = \frac{r_3}{r_{I_{\Delta T(\eta)}}} \le \varepsilon,$$

then the **relaxed type-M function dependency** during ***ΔT(η)***:

$$\boldsymbol{X_{I(d_1, \tau')} \xrightarrow{\Psi(\Delta T(\eta), X_I, Y_I) \le \varepsilon} Y_{I(d_2, \tau')}}$$

holds, where $\Psi(\Delta T(\eta), X_I(v_k), Y_I(v_k))$ is ***ΔT(η)*- *DVR*-$\boldsymbol{v_k[X_I, Y_I]}$**, $r_{I_{\Delta T(\eta)}}$ is the combinatorial number of any pair of attributes in X_I or Y_I during $\Delta T(\eta)$, $r_3 \subseteq r_{I_{\Delta T(\eta)}}$ is the number of the DVEs$-X_I, Y_I$ during $\Delta T(\eta)$, m_1 is the number of the tuples during $\Delta T(\eta)$, $d_1 \in D[X]$, $d_2 \in D[Y]$, and $\tau', \tau'', \varepsilon \in [0,1]$ are thresholds.

Remark 4: If $X_{(d_1, \tau')} \xrightarrow{\Psi(\Delta T(\eta), X, Y) \le \varepsilon} Y_{(d_2, \tau'')}$ doesn't hold, and there are ANEs in X and Y, we can delete some $v_I[X]$s and $v_I[Y]$s corresponding to them from X and Y, then we get X_I and Y_I, and we have $X_{I(d_1, \tau')} \xrightarrow{\Psi(\Delta T(\eta), X_I, Y_I) \le \varepsilon} Y_{I(d_2, \tau'')}$ holds. This means the case of definition 14 is happening.

5. Event-Based Data Analysis Example

The following records represent a person's meditation input data including EEG from brainwave headset and GazeX and GazeY from the smart phone:

Time		EEG	GazeX	GazeY
2018-2-20	16:57:00	53	0.02884405	0.36825011
2018-2-20	16:57:01	57	-0.0057313	0.39013446
2018-2-20	16:57:02	74	0.00372011	0.33091585
2018-2-20	16:57:03	84	0.07300814	0.36468598
2018-2-20	16:57:04	90	0.06822054	0.39343803
2018-2-20	16:57:05	84	0.01829791	0.35769521
2018-2-20	16:57:06	74	0.07686714	0.4012554
2018-2-20	16:57:07	43	0.05864623	0.40079645
2018-2-20	16: 57:08	27	0.08833459	0.41172976
2018-2-20	16: 57:09	43	0.02981886	0.40139946
2018-2-20	16: 57:10	43	0.08068578	0.3896068
2018-2-20	16: 57:11	67	0.07305756	0.37007838
2018-2-20	16: 57:12	77	0.05570461	0.44665981
2018-2-20	16: 57:13	70	0.05092989	0.44977627
2018-2-20	16: 57:14	67	0.03441077	0.41223145
2018-2-20	16: 57:15	69	0.03749303	0.49343493
2018-2-20	16: 57:16	67	0.03365155	0.42283732
2018-2-20	16: 57:17	61	0.0471089	0.47274698
2018-2-20	16: 57:18	54	0.04033958	0.48874432
2018-2-20	16: 57:19	56	0.04615196	0.45340732
2018-2-20	16: 57:20	60	0.08277113	0.43117775
2018-2-20	16: 57:21	75	0.12389434	0.4264601
2018-2-20	16: 57:22	90	0.02021705	0.47553028
2018-2-20	16: 57:23	90	0.04613996	0.37573326

Firstly we define: for any attribute X,

$$d_{\max\_X_{ij}} = \max_{1 \le i, j \le m} |v_i[X] - v_j[X]|$$

denotes the maximum of the distance between the values of any two tuples v_i, v_j in the attribute X, where m is the number of the tuples during the whole time T, and

$$d\left(v_i[X], v_j[X]\right) = \frac{|v_i[X] - v_j[X]|}{d_{\max\_X_{ij}}}$$

is the distance function.

Then according to the above distance function, for the attribute EEG, for simplicity we denote it as E, we have

$$d\left(v_i[E],v_j[E]\right)=\frac{|v_i[E]-v_j[E]|}{d_{\max\_E_{ij}}},$$

$$d_{\max\_E_{ij}}=\max_{1\le i,j\le m}|v_i[E]-v_j[E]|.$$

It is easy to see from the table that $d_{\max\_E_{ij}}=90-43=47$.

Assuming we can choose the time from 16:57:00 to 16:57:07 on February 20th, 2018. For any pair of tuples during this time, we calculate their distance functions as follows:

$$d\left(v_1[E],v_2[E]\right)=\frac{|53-57|}{47}=\frac{4}{47}\approx 0.0851.$$

Similarly, we can get

$d\left(v_2[E],v_5[E]\right)\approx 0.7021$, $d\left(v_3[E],v_6[E]\right)\approx 0.2128$,

$d\left(v_4[E],v_7[E]\right)\approx 0.2128$, $d\left(v_5[E],v_8[E]\right)=1$,

$$\vdots$$

Obviously, if $\tau'=0.7$, then except that

$$d\left(v_1[E],v_5[E]\right)\approx 0.7872>\tau',$$
$$d\left(v_2[E],v_5[E]\right)\approx 0.7021>\tau',$$
$$d\left(v_4[E],v_8[E]\right)\approx 0.8723>\tau',$$
$$d\left(v_5[E],v_8[E]\right)=1>\tau',$$
$$d\left(v_6[E],v_8[E]\right)\approx 0.8723>\tau',$$

for the other pair of tuples, $d\left(v_i[E],v_j[E]\right)\le\tau'$ $(i,j=1,2,\cdots,8,\ \ i\ne j)$ always holds.

And for the attribute GazeY, for simplicity we denote it as GY, then

$$d\left(v_i[GY],v_j[GY]\right)=\frac{|v_i[GY]-v_j[GY]|}{d_{\max\_GY_{ij}}},$$

$$d_{\max\_GY_{ij}}=\max_{1\le i,j\le m}|v_i[GY]-v_j[GY]|,$$

and

$$d_{\max\_GY_{ij}}=0.4012554-0.33091585=0.07033955.$$

Thus we can similarly get their distance functions:

$$d\left(v_1[GY],v_2[GY]\right)=\frac{|0.36825011-0.39013446|}{0.07033955}\approx 0.3111,$$

and

$$d\left(v_2[GY],v_3[GY]\right)\approx 0.8419,$$
$$d\left(v_3[GY],v_5[GY]\right)\approx 0.9017,$$
$$d\left(v_4[GY],v_6[GY]\right)\approx 0.0994,$$
$$d\left(v_5[GY],v_8[GY]\right)\approx 0.0918,$$
$$\vdots$$

If $\tau''=0.6$, then except that

$$d\left(v_2[GY],v_3[GY]\right)\approx 0.8419>\tau'',$$
$$d\left(v_3[GY],v_5[GY]\right)\approx 0.9017>\tau'',$$
$$d\left(v_3[GY],v_7[GY]\right)=1>\tau'',$$
$$d\left(v_3[GY],v_8[GY]\right)\approx 0.9935>\tau'',$$
$$d\left(v_6[GY],v_7[GY]\right)\approx 0.6193>\tau'',$$
$$d\left(v_6[GY],v_8[GY]\right)\approx 0.6128>\tau'',$$

for the other pair of tuples, $d\left(v_i[GY],v_j[GY]\right)\le\tau''$ ($i,j=1,2,\cdots,8,\ \ i\ne j$) always holds.

It is clear that $v_2[E]\cong_{(d_1,\tau',t_i,t_j)}v_3[E]$ holds, whereas $v_2[GY]\cong_{(d_2,\tau'',t_i,t_j)}v_3[GY]$ doesn't holds. By the definition 3, this is a dependency violation event (DVE). In fact, $\Delta T(\eta)$- DVEs -[E, GY] are as follows:

$$\Delta T(\eta)\text{ - DVEs -}v_2,v_3[\text{E, GY}],$$
$$\Delta T(\eta)\text{ - DVEs -}v_3,v_5[\text{E, GY}],$$
$$\Delta T(\eta)\text{ - DVEs -}v_3,v_7[\text{E, GY}],$$
$$\Delta T(\eta)\text{ - DVEs -}v_3,v_8[\text{E, GY}],$$
$$\Delta T(\eta)\text{ - DVEs -}v_6,v_7[\text{E, GY}].$$

Therefore

$$\Psi(\Delta T(\eta),E,GY)=\frac{5}{28}\approx 0.1786.$$

If $\varepsilon=0.18$, then

$$\Psi(\Delta T(\eta),E,GY)\le\varepsilon.$$

And according to the definition 6, as long as $v_i[E]\cong_{(d_1,\tau',t_i,t_j)}v_j[E]$ holds, there is almost $v_i[GY]\cong_{(d_2,\tau'',t_i,t_j)}v_j[GY]$ holds. So

$$E_{(d_1,\tau')}\xrightarrow{\Psi(\Delta T(\eta),E,GY)\le\varepsilon}GY_{(d_2,\tau')}$$

holds.

We note that for the tuple v_3, according to definition 10, we have

$$\Psi(\Delta T(\eta),E(v_3),GY(v_3))=\frac{4}{7}\approx 0.5714>\varepsilon,$$

i.e., v_3 constitutes an abnormal event (ANE).

At the same time, we can get

$$\Psi(\Delta T(\eta),E(v_1),GY(v_1))=\frac{0}{7}=0\le\varepsilon,$$

$$\Psi(\Delta T(\eta),E(v_2),GY(v_2))=\frac{1}{7}\approx 0.1429\le\varepsilon,$$

$$\Psi(\Delta T(\eta), E(v_4), GY(v_4)) = \frac{0}{7} = 0 \le \varepsilon,$$

$$\Psi(\Delta T(\eta), E(v_5), GY(v_5)) = \frac{1}{7} \approx 0.1429 \le \varepsilon,$$

$$\Psi(\Delta T(\eta), E(v_6), GY(v_6)) = \frac{1}{7} \approx 0.1429 \le \varepsilon,$$

$$\Psi(\Delta T(\eta), E(v_7), GY(v_7)) = \frac{2}{7} \approx 0.2857 > \varepsilon,$$

$$\Psi(\Delta T(\eta), E(v_8), GY(v_8)) = \frac{1}{7} \approx 0.1429 \le \varepsilon.$$

Therefore there is no ANE-1 happening during the time from 16:57:00 to 16:57:07 on February 20th, 2018. Obviously, there is also ANE-2 appearing.

It is clear during the time from 16:57:00 to 16:57:07 on February 20th that $E = \left(v_1[E], \cdots, v_8[E]\right)$, $GY = \left(v_1[GY], \cdots, v_8[GY]\right)$. According to the above calculation process, we know if $\varepsilon = 0.05$, then

$$E_{(d_1, \tau')} \xrightarrow{\Psi(\Delta T(\eta), E, GY) \le \varepsilon} GY_{(d_2, \tau')}$$

doesn't hold. However, for

$E_{\mathrm{I}} = \left(v_1[E], v_2[E], v_4[E], v_5[E], v_6[E], v_7[E], v_8[E]\right) \subseteq E$,

$GY_{\mathrm{I}} = \left(v_1[GY], v_2[GY], v_4[GY], v_5[GY], v_6[GY], v_7[GY], v_8[GY]\right)$
$\subseteq GY$,

there is only one dependency violation event (DVE): $\Delta T(\eta)$ - DVEs -v_2, v_3[E, GY].

Then we can get

$$\Psi(\Delta T(\eta), E_{\mathrm{I}}, GY_{\mathrm{I}}) = \frac{1}{21} \approx 0.0476 < \varepsilon.$$

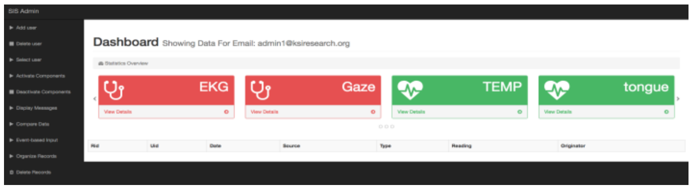
So $E_{\mathrm{I}(d_1, \tau')} \xrightarrow{\Psi(\Delta T(\eta), E_{\mathrm{I}}, GY_{\mathrm{I}}) \le \varepsilon} GY_{\mathrm{I}(d_2, \tau')}$. This is the case of Definition 14.

6. User Scenarios

In TDR system, sensor data from different devices, devices like temperature, humid, gaze, and etc, will be stored in a heap. In order for the administrator to better organize those data into separate relations, we have developed some tools to facilitate the process. The following are the steps how an admin can manage the system.

6.1. Scenario One: Organize Records

Upon login as an admin, you can see the following:



Dashboard Showing Data For Email: admin1@ksiresearch.org

Figure 9. The Dashboard.

To write data into different relations, click organize records, then you can choose which relation you wish to write the data to.

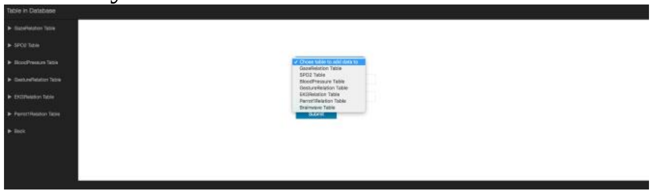
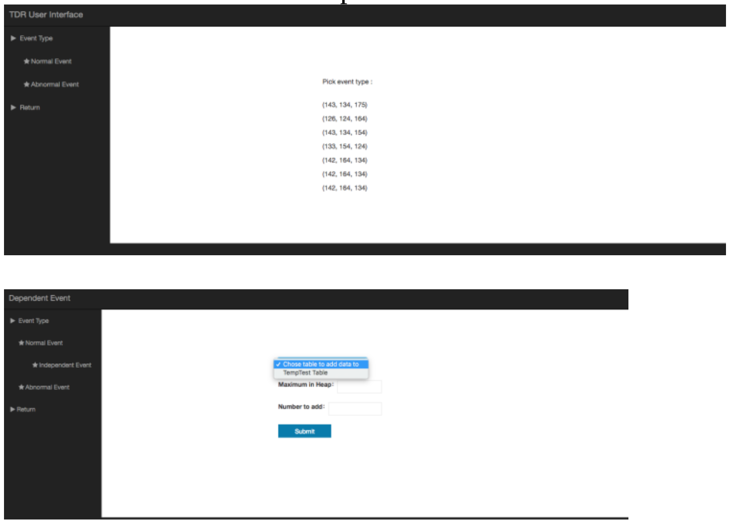


Figure 10. Choose relations.

Upon selecting which relation the admin prefer to write data to, the system will show how many records are available in heap. The admin may type in the number of records he/she wants to write into the specific relation, but the number has to be no greater than maximum records in heap, after click on submit, the system will remind the admin whether his/her action was preformed successfully.

6.2. Scenario Two: Event-based Input

From the main page, if admin wish to move data to relations subject to certain restrict,ion he/she may choose to use event-based input tool.



Pick event type :

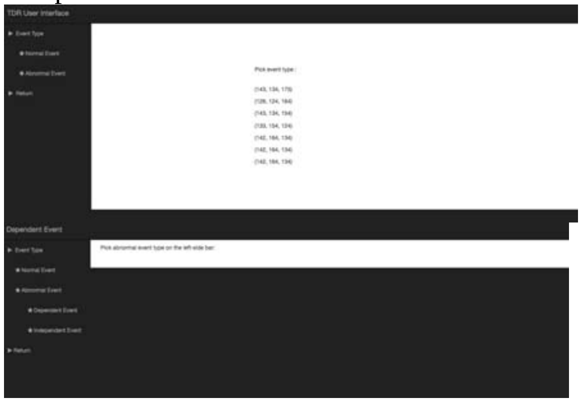
(143, 134, 175)
(126, 124, 164)
(143, 134, 154)
(133, 154, 124)
(142, 164, 134)
(142, 164, 134)
(142, 164, 134)

Maximum in Heap :
Number to add :

Figure 11. Normal event.

After admin has chosen normal event, admin can then pick which relation he/she wish to choose. If a tuple is a normal event for the relation, then the tool can add the tuple into the relation. Similarly, after chose abnormal event, the tool will prompt admin to pick which event (aka: dependent event or independent event) he/she want to add records to.

We will give an example upon picking dependent event, but the flow will be the same if admin chose independent event.



Pick event type :

(143, 134, 175)
(128, 124, 164)
(143, 134, 154)
(133, 154, 124)
(142, 164, 134)
(142, 164, 134)
(142, 164, 134)

Pick abnormal event type on the left side bar:

Figure 12. Independent event.

After chose which relation data admin wish to apply algorithm on, the tool will select data records and apply (2) on it, if data records satisfy (2), then move it to the correspond relation.

7. Discussion

In this paper we describe an experimental TDR system with the following features: 1) the experimental system can run on a smart phone and therefore portable; 2) a meditation validation channel to check the consistency between the predictions via gaze features vs. features to increase the accuracy of meditation prediction; 3) through event-based data input, modeling and analysis, a user can access the brainwave from a one-channel NeuroSky Mindwave headset and gaze data from a Samsung phone and the consistency check graph via a web GUI; 4) QA and rating, where a user can provide feedback right after his/her meditation process, master/teacher will rating the meditation quality based on such feedback and previous measurement data. We can also track user's typing movements when providing feedback to measure the users' muscle change during and after meditation.

An initial experiment was designed and conducted to test the ability of monitoring meditation state via brainwave and gaze tracking techniques, as well as observe the relationship between the two sources of signals. Preliminary results indicated a trend of positive relationship (correlation coefficient = 0.982) between gaze y-axis signals and brainwave signals (Figure 13), which indicates the validity of our approach in meditation detection as well as inspired us to further investigate their degrees of correlations.

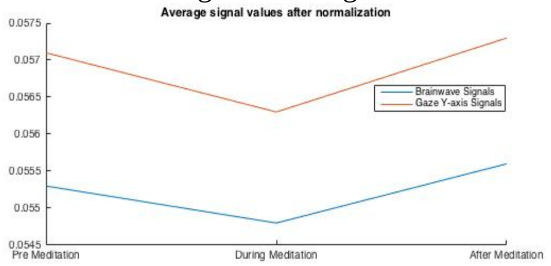


Figure 13. Preliminary results on meditation states tracking via brainwave signals and gaze signals.

The current system has certain limitations: 1) headset requires a precise wearing process to extract sensor data, otherwise a portion of the data may be missing. Users who are not professional enough or without external support, will only have partial data, which is less accurate; 2) Gaze tracking via front facing camera of smart phone is portable and maneuverable, but lack of accuracy due to the noisy luminance effect in real environment as well as the user's meditating habit.

For future work, we need to develop techniques to overcome the above mentioned limitations, as well as to design approaches to help people better understand their meditation state without too much manual intervention. More experiments need to be designed and carried out to validate the proposed approach.

Acknowledgement

This research was supported in part by Knowledge Systems Institute, USA. The research of CuiLing Chen was supported by the Visiting Scholarship Fund of Education, Department of Guangxi Zhuang Autonomous Region, P R China.

References

[1] Shi-Kuo Chang, JunHui Chen, Wei Gao and Qui Zhang, TDR System - A Multi-Level Slow Intelligence System for Personal Health Care, SEKE2016, Hotel Sofitel, Redwood City, CA, USA, 183-190.

[2] Shi-Kuo Chang, Wei Guo, Duncan Yung, ZiNan Zhang, HaoRan Zhang and WenBin You, A Mobile TDR System for Smart Phones, DMSVLSS 2017, Wyndyam Pittsburgh University Center, Pittsburgh, PA, USA, 75-85.

[3] https://en.wikipedia.org/wiki/User_interface_design

[4] S.-K. Chang, V. Deufemia, G. Polese, and M. Vacca, A normalization framework for multimedia databases, IEEE Trans. Knowl. Data Eng., vol. 19, no. 12, pp. 1666–1679, Dec. 2007.

[5] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese, Relaxed Functional Dependencies - A Survey of Approaches, IEEE Transactions on knowledge and data engineering, VOL. 28, NO. 1, JANUARY 2016.

A Logic User-Based Algorithm to Improve Node Distribution in Wireless Sensor Network

Balzano Walter
Dip. Ing. Elettrica e Tecnologie dell'Informazione
Università di Napoli, Federico II
Napoli, Italy
Email: walter.balzano@gmail.com

Stranieri Silvia
Università di Napoli, Federico II
Napoli, Italy
Email: silviastranieri1047@gmail.com

***Abstract*—Localization in Wireless Sensor Networks (WSN) is a largely discussed research topic. Different solutions have been proposed to solve the localization problem over the time, exploiting techniques such as angle-based, range-based, and range-free. In our previous work, we proposed a logic range free algorithm able to uniformly distribute hubs around a given environment: this distribution is aimed to give the best coverage possible of the areas of interest. In this procedure, the presence of obstacles is taken into account, since it can impact the signal power: for this reason, an attenuation factor has been introduced to understand in which measure the obstacles modify the result. The algorithm is based on Prolog backtracking technique, which reflects the procedure of organizing the relative positions of nodes at each step, in order to optimize their distribution over the environment. The main goal of this work is to improve this approach, by considering not only the need to ensure a certain signal in each zone, but also to focus on the areas where a big usage from the clients is detected, in order to prevent the network saturation, without losing the coverage property. To this purpose, a displacement factor is introduced to vary the previous result in favor of a user-based distribution. The value of this factor has to reflect the optimum trade-off between coverage and user's usage need.**

Keywords: WSN, localization, coverage, Prolog.

I. Introduction

A Wireless Sensor Network is made by nodes self-organized, broadcasting information and data all over the net. Due to the huge potential of such a network, many proposals have been made to improve the communication between the nodes, to select the best routing protocol possible, and to locate nodes inside the net. To find the best coverage possible for the area of interest, the localization technique used is an important aspect in research field.

The absence of obstacles makes the hub positioning in outdoor environment a not interesting problem, while in case of presence of walls, doors, or other obstacles, the positioning of nodes in a network can became a bottleneck in the localization field. Any kind of impediment can alter the power of the transmission signal, and this is something we want to deal with in order to guarantee an optimal distribution of hub in indoor environment too.

In our previous work [22]. We designed a range-free algorithm able to provide a map of the optimal hubs distribution over a given environment, in such a way that the best coverage possible is ensured. The approach is the logic one, since the way the algorithm works is based on backtracking: whenever a hub is added into the environment, its total mapping is arranged, and the position of each hub is computed again in order to optimize some metric on the signal. This approach provides a non-greedy algorithm whose solution is guaranteed to be the global optimum, rather than the local one. The provided algorithm presents two variants of execution, that differ in the choice of having *anchor* nodes or not.

Aware of the need to cover as much as possible the interest area, this approach does not consider the user's need: often, a given environment has not an equally distributed usage, but a more powerful signal can be needed in a room rather than another. For instance, if we consider an environment made by rooms, and one of these rooms is a pretty big warehouse, the first approach of our algorithm probably would place two hubs to cover that room but, by analysing the user's usage information turns out that the warehouse signal is never used: it's clear that using two hubs to cover a non-used area is a waste of resources, and in this work we want to avoid this situation. The aim is to consider the user's usage as an important parameter for the hub distribution, but not the only one: indeed, we do not want to lose the coverage property.

The rest of the work is organized as follows: Section II contains important aspects of literature that inspired our work, and other related research concerning localization issues in wireless sensor networks; in Section III, we explain the motivations that let us improve our previous work, by first recall all the main features of it, and then summarize the aspects that can be improved in our perspective; Section IV contains the very strategy to introduce the user's usage information in the approach, and how it is integrated with the coverage request in order to obtain the best possible solution that respects both the properties; finally, in Section V, the conclusions of this work and its advantages are shown.

II. Related Work

Wireless Sensor Networks are, nowadays, one of the most studied research topics. The development of such networks was initially born for military purposes, while now, as explained in [2], there is a bunch of applications of these nets: environment

DOI reference number 10.18293/VLSS2018-031

and structures monitoring, traffic management, surveillance, and many other application fields. Actually, this is the reason why many studies are made about this topic and all the related issues, such as localization of nodes in such a network and signal distribution. An important indicator which is largely used in Wireless Sensor Networks for localization purposes is the RSSI (Received Signal Strength Indicator). This indicator provides useful information about the signal power for any retrieved hub in the environment. For instance, in [6] RSSI is exploited in traffic control field in order to estimate the positioning of vehicles. They state that Global Positioning System does not always guarantee the accuracy needed in cooperative-vehicle-collision-warning systems, while the radio-based-ranging approach founded on RSSI improves the accuracy. Using the same approach, in [7] they propose a range-free algorithm based on RSSI comparisons, called Ring Overlapping. Each node uses overlapping rings in order to guess the possible area in which it lies: given an anchor node *A*, each ring is actually generated by comparing the RSSs received by a node from *A* and the ones received by other anchor nodes from *A*. Even in [20], they highlight the importance of positioning accuracy in vehicle-to-vehicle field.

A crucial variation point in localization algorithms in WSN is in the choice of using anchor nodes or not. In [3] is proposed an anchor-based localization approach: the main idea is that each anchor is aware of its position, because equipped with GPS, and it periodically shares its current location with the other nodes which are able, thanks to this information, to locate themselves. This approach tolerates the presence of obstacles and has the benefit of not requiring any hardware modification. Oppositely, in [12] they prefer an anchor-free approach, summarizing all the drawbacks of having fixed nodes in a network.

In our previous work, we focus on logic strategies in order to deal with many problems related to traffic control, such as in [15], sometimes integrating it with clustering techniques ([14], [16]), or Distance geometry problem, like in [17]: even in this work we use the logic approach (*i*) to facilitate the comprehension of the algorithm behaviour, through elegant and compact code, and (*ii*) to exploit the expressiveness power of Prolog and its cut operator to prune useless computational paths. But, many other localization techniques are proposed in literature. In particular, in [4] they highlight three categories of localization approaches: (*i*) AOA (Angle of Arrival) represents the angle between the propagation direction and some reference direction (orientation) and it constitutes the information which is exchanged between nodes, so that their localization can be performed by using trilateration [8], (*ii*) Distance Related Measurements, and (*iii*) RSS (Received Signal Strength) profiling. Moreover, in [5] they propose an indoor localization approach, called EZ localization algorithm which estimates the positioning of 2D point in terms of absolute coordinates: latitude and longitude.

The main inspiration for this work is given by our previous work [22], with the aim to improve it by considering an important metric not taken into account by now, which is the usage of the network that the client typically does.

First, we introduce Wireless Sensor Networks, a system of nodes which exchanges data wirelessly. All this information can be possibly held and elaborated by a control unit. As known, each net can have a particular topology, which characterizes the behaviour of its component. In [1], they summarize essentially six kinds of network topologies:

1. Star topology: each node is connected to a single hub which filters any communication;
2. Ring topology: there isn't a leader, the information exchange follows one direction (the one of the ring);
3. Bus topology: there is a communication channel were all the information passes through;
4. Tree topology: hierarchical structure is the base of any communication;
5. Fully connected topology: each node is connected to any other node and this makes this topology suffer from NP-complexity;
6. Mesh topology: nodes have a regular distribution and each node communicates with its nearest neighbour.

Another important ingredient concerning localization is the Received Signal Strength Indicator (RSSI). This indicator provides the power of the received signal in a certain point and it has a strong relevance since not only it gives important knowledge for the purpose of localization, but it is also recognizable by any device on the market. For instance, WirelessNetView is an application which freely provides the percentage of the received signal by any retrieved hub.

We present the most famous approaches to estimate the position of a point in a Wireless Sensor Network. The initial classification we can introduce divides localization techniques in anchor-based and anchor-free: in the first approach, the network presents some special nodes, the anchors, which are aware of their position since they are equipped with a Global Positioning System, while all the other nodes, the targets, guess their location with respect to the anchors one; while someone actually prefers this kind of approach, such as in [13], some other authors have found some limitations in anchor-based algorithm, hence an anchor-free approach has been introduced. For instance, in [12] they suggest this kind of approach, since they indicate three reasons why the anchor-base algorithms are not the best choice: (*i*) there is a waste of time due to the manual insertion of anchor nodes; (*ii*) anchor-based algorithms are unstable, since a small mistake in the anchors positioning may cause a huge mistake in the wireless sensor network final configuration; (*iii*) anchor-based algorithms are not scalable.

III. Motivations

In our previous work [22], the localization problem is solved by means of a simulation program that has the aim to distribute nodes in a given environment according to a coverage criterion: we consider three possible node distributions:

- Random;
- Geometric;
- Signal-based.

Following a random distribution, nodes are placed randomly all over the environment, without any kind of optimization criteria.

With a geometric distribution, instead, the program tries to place the nodes in a geometric way in the environment, according to its the shape.

In the third case, with a signal-based distribution, the nodes are placed trying to optimize the signal spread over the environment. In this case, each insertion of a node in the area puts in doubt the previous placements if the signal could have been distributed in a better way.

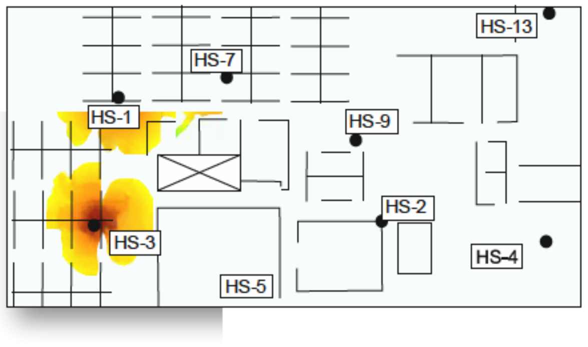


Figure 1: Generic fingerprinting radio-map with different nodes and variation of signal due to presence of obstacles

In any of these cases, the main goal is to obtain the best coverage possible in the environment, in such a way that it's not possible to find areas where the signal is not available (even if with a weak power). It focuses on *range-free* approach in order to provide a localization algorithm. This kind of localization technique uses some particular maps, called *fingerprinting radio-maps* where the signal power of each node is represented by its fingerprint (fingerprint-based techniques have been used in [19] and [21], too). Since these maps are created through measurements of the retrieved signal in various points of the environment, the presence of obstacles has an impact on the signal power, as we can observe in Figure 1, where the variation of colour intensity reflects the signal attenuation. Each device is represented by the black areas, and as the distance from it increases the strength of the signal decreases: this is expressed in a colour variation from dark red to yellow. Moreover, we can observe that this variation is not regular nearby the obstacles (represented by the black lines): in fact, the presence of obstacles deforms the signal and lets the color turn into yellow more quickly (such as in the case of hub 2).

For each point the power level is computed by using the inverse-square law:

$$P = \frac{P_M}{(x_i - x_n)^2 - (y_i - y_n)^2}$$

where (x_i,y_i) are the coordinates of one of the point, (x_n,y_n) are the coordinates of one possible point in the environment, P_M is the maximum signal for the node, and P is the computed signal power for that point with respect to that node. This law tells us that the signal power is inversely proportional to the square of the distance.

Our simulation program works as follows: initially, it provides an environment in which we can put nodes and obstacles; then, it creates a list of points which represent the grid where we are going to simulate the retrieved power. For each obstacle, it picks a point and it generates for it the dimensions and an attenuation factor α which indicates how that obstacle influences the signal. In the end, a vector for each obstacle is obtained. Now, nodes have to be located and the program can do this by following the three different node distributions cited above.

Subsequently, a vector for each point is generated, where each component represents the simulated signal power from a node of the network. By building the union of a specific component taken by all the points, we obtain a fingerprinting radio-map.

Step by step, the program chooses where is preferable adding devices in order to have the best coverage possible, as explained in [22]. The program, as usually happens, places the first device in the middle of the hole environment, in order to have a good signal distribution. At the very beginning, it tries to cover the biggest area of the environment, as we can observe from Figure 2.

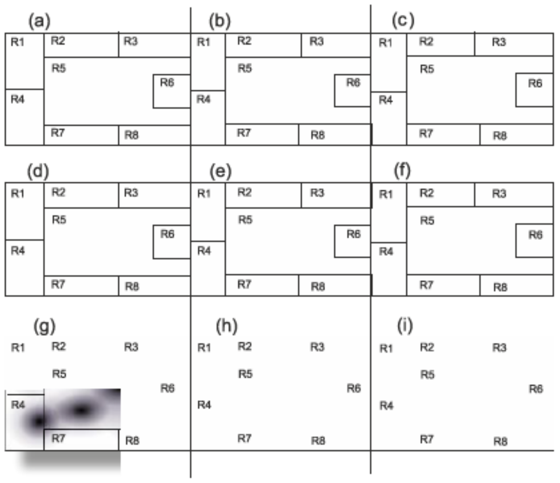


Figure 2: Evolution phases of the simulation program execution over an environment with some obstacles

In 4.c, the program decides to add a hub in one of the smaller rooms to have an improvement of signal distribution. After having covered the whole central area (Figure 4.d), the program starts adding devices in all the other rooms (Figures 4.e, 4.f, 4.g and 4.h), until it reaches the coverage of the entire environment and stops, as shown in Figure 4.i.

Moreover, we can see how the presence of obstacles determines a distortion in the signal shape, which is proportional to the attenuation factor α. This example is without any anchor nodes, but it is still possible adding some fixed node before the simulation start: in this case, clearly the addition of other hubs wouldn't have affected the position of anchors and thus it is likely that the final configuration of the network would have been different from the obtained one.

In some cases, a more meaningful criterion can be found to distribute nodes around a given environment, by considering the client needs: for instance, if we consider an environment made of some rooms, and one of them is actually a warehouse of big dimensions, but where very few signals of usage retrieved, our algorithm probably would place a certain number of hubs in that room in order to cover it, but this could be a redundancy if we take into account the user's usage. Indeed, some of the hubs used for the warehouse could have been better employed in areas where the signal is needed the more.

We analyzed the behavior of our algorithm, and the result highlights that its precision can vary according to number of nodes, grid dimension, and environment shape. In particular, the more the nodes are the more the precision of the algorithm grows. This does not mean that we can increase the number of nodes in an unchecked way, since we couldn't obtain an absolute precision: this is a consequence of the fact that measurements are made in map points which are in the detections grid too. This research gives as result all the points of the grid that are close to the point we are looking for.

Concerning the grid features, by increasing grid dimensions the precision increases. This is obvious, since there is a higher probability that points are close to the one we are looking for, during the comparison phase.

Finally, as we could expect, the more the environment grows the more the error increases, since each node influences just a small part of the entire environment, hence localization mistakes are more frequent.

Our final consideration is that the random distribution should be avoided, since it leads to less precise results; oppositely, both geometric and signal-based distributions provide solutions with a good precision, hence should be preferred to the random one.

Clearly, this approach can impact the coverage requirement: in some cases, the distribution of hubs according to the user's usage can decrease the environment coverage, by having some small areas not covered at all. But, the guess is that those not covered areas are surely not of interest for the users, and it is reasonable to reduce the coverage in not used areas in favour of those with a high density of users. In a border case, we could have whole rooms not covered at all, but this is not the goal of our improvement, since this would be a too heavy restriction. For this reason, we introduce a displacement factor δ that represents the percentage of how much the user's distribution influences the hub's positioning based on environment coverage. According to this factor, δ=100% represents the border case explained above, where the distribution proposed by our starting algorithm, based on coverage, is ignored in favour of a distribution based exclusively on user's usage. The opposite case is given by δ=0% that is essentially the same algorithm proposed in [22], where the only parameter, that is taken into account, is the environment coverage.

In a generic case, with δ=50% for instance, the idea is that the first resulting distribution given by our simulation algorithm is slightly modified in order to improve the user's usage, by not losing the property of coverage guaranteed by the first approximation.

This improvement of our first approach [22] can seriously lead to a more efficient nodes distribution that provides not only the signal coverage in the whole environment, but also a stronger signal power where needed for the users, by finding the best value for the displacement factor.

In the next section, we are going to describe the mechanism that allows us to retrieve and exploit the user's usage statistics.

IV. User Usage Compensation

The information about how much a certain area is used by network's clients can be easily detected by using some utilities able to retrieve statistics concerning the usage during the time, such as *WirelessNetView* for the RSSI (Received Signal Strength Indicator) data detection.

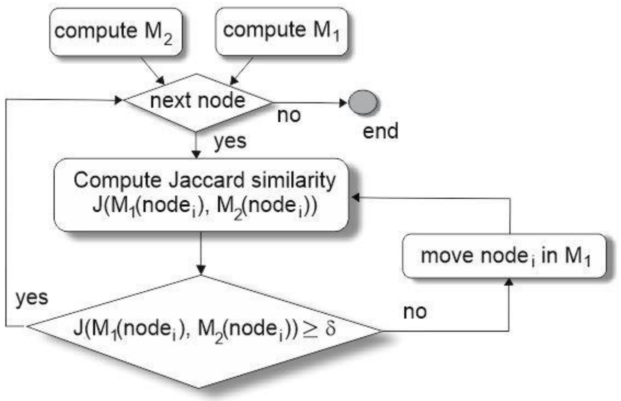

compute M$_2$
compute M$_1$
next node
no
end
yes
Compute Jaccard similarity J(M$_1$(node$_i$), M$_2$(node$_i$))
move node$_i$ in M$_1$
yes
no
J(M$_1$(node$_i$), M$_2$(node$_i$)) ≥ δ

Figure 3: flow of control with the user's usage statistics compensation with the aim to obtain a fixed minimum displacement with respect to the original solution based on coverage

The idea is to exploit such information in order to build a map according exclusively to the user's usage of the network, similarly to how we build the map based on environment coverage. This is a border map, representing a displacement factor of 100%, which is not our aim, but it is useful in order to obtain the desired displacement value δ. After the generation of the map based on coverage criterion, a second map has to be created, considering only the user's usage. These two maps represent the two opposite limits (δ=0% and δ=100%), but what we are going to compute is a third map that is the result of a trade-off between coverage and user's usage, with a displacement δ in the interval (0,100).

In order to do that, we indicate with M_1 the map based on coverage and with M_2 the one based on user's usage. We will exploit a largely used similarity measure, which is the Jaccard index. This indicator represents the similarity between the corresponding hubs in M1 and M2, by taking the intersection between the areas covered by the nodes and normalizing it with the union.

$$J(M_1(node_i), M_2(node_i)) = \frac{coverage(M_1(node_i))}{coverage(M_2(node_i))}$$

Where $M_1(node_i)$ represents the i-th hub in the first map, and $coverage(M_1(node_i))$ is the set of space points reached by the signal of the i-th node in the first map. Clearly, the Jaccard index is in the interval [0,1] (where 0 is a non-similarity indicator, and 1 is the maximum similarity possible). Clearly, we are not interested in obtaining a unitary similarity between the two maps, since this would possibly determine uncovered areas of the environment and this is not the best approach. What we want to obtain is a similarity based on the displacement value previously fixed, in particular:

$$J(M_1(node_i), M_2(node_i)) \geq \frac{\delta}{100}$$

Thus, the aim is to move a node from the first map M_1 in order to let it similar to the same node, but in the second map M_2, in such a way that their similarity is proportional to the displacement factor. Going through successive approximation of the map that keep moving the node toward the ideal position (according to the user's usage), the procedure stops when the minimum desired similarity is reached.

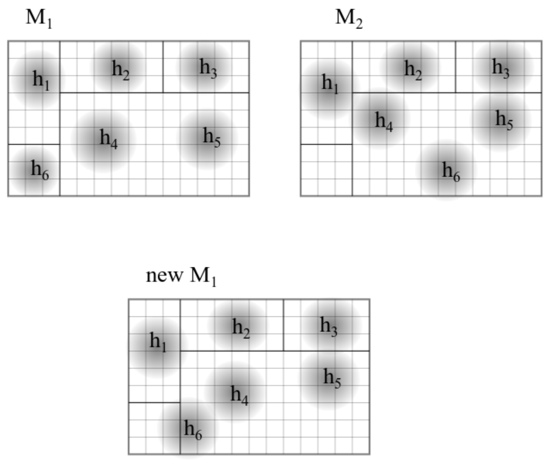


Figure 4: example of how the two border maps are arranged to obtain a new map, which is a trade-off between coverage and usage, considering a displacement of 30%

In Figure 3 is show the procedure explained by now: both the maps are generated: the first one only considers the coverage aspect, while the second one is based on the usage information of the network. The next steps are repeated until all the nodes of the network are considered. Each node is compared with the same node of the other map in order to compute the Jaccard similarity between them, according to the ratio between the points in the intersection of the nodes range and the ones in the union. Finally, the node is moved in the first map until the minimum desired displacement factor δ is obtained as Jaccard similarity between the two corresponding nodes that are being analysed. At the end of this procedure, the map M_1 will no longer reflect the nodes distribution aimed to maximize the environment coverage, but it will be the modified distribution that takes into account the usage of the network too, in a percentage induced by δ.

In Figure 4, an example of the possible result is shown: let's consider the square as unity of measure, and the box surrounding the hub (9 units in total) the points in the range of the corresponding hub. The map M_1 shows the hub's distribution according to the environment coverage, while the map M_2 the optimal distribution based on information detected concerning the usage of the network. In this example, we consider δ=30%, meaning that for each pair of corresponding hubs we need a Jaccard similarity of at least 0.3. Intuitively, the hubs h_1, h_2 and h_3 respects this similarity, thus we focus on the remaining three hubs h_4, h_5 and h_6.

Looking at the hub h_4 in M_1 and M_2, we can see that their intersection is 4 (the number of squares that they share), while the union is 18 (twice the number of squares in the range, that is 9 as we said): this means that the Jaccard similarity between them is 4/18=0.2, which is not enough with respect to δ. Indeed, in the new map M_1 the hub h_4 is moved in such a way that the similarity reaches the desired one.

The hubs h_5 in M_1 and M_2 have an intersection of 6 points, and the union is always 18, as before. Thus, the Jaccard similarity is 6/18=0.3, which is acceptable, and this is the reason why it is not moved from the starting position in M_1.

Finally, hubs h_6 no intersection, indeed the movement is more evident with respect to h_4 since it was not similar at all with its corresponding in M_2.

The final configuration is the one shown in new M_1, where all the similarities between corresponding hubs is at least 0.3.

Procedure: usage compensation

```
1: map1(node,position).
2: map2(node,position).

3: check_jaccard(node,δ):-
4:     J(map1(node,p1),map2(node,p2)>=δ/100,!.

5: check_jaccard(node,δ):-
6:     J(map1(node,p1),map2(node,p2))<δ/100,
7:      move(node,map$_1$),check_jaccard(node,$\delta$).
```

The procedure shown in the code fragment above shows exactly the way this approach works: let's suppose of having a Prolog fact for each node of the maps that says the position of that node in the corresponding map.

The predicate *chech_jaccard(node,δ)* checks if the Jaccard similarity is at least the minimum desired one (δ/100), otherwise the corresponding node is moved in the map M_1 and the Jaccard similarity is computed again, until the constraint is satisfied.

In this work we deal with localization problem by using a logic programming language: we can see how the logic approach and Prolog programming language help us avoiding redundancy in computation. This is made thanks to the cut operator (!) that as soon as an advantageous computation branch is found, discards the other paths, in order to not analyse branches that would have led to useless solutions.

It is clear that without any further check, this successive approximation of nodes position inside the map M_1 can progressively let a room to be free of any kind of signal: this is what we wanted to avoid in our starting considerations. To avoid this kind of behaviour, we have to check that the next node movement does not impact the coverage of a whole room, before it is performed. If this happens, we have two possibilities:

- Reduce the displacement factor δ in order to relax the minimum required similarity;
- Increase the resources: add a new hub in the room that becomes free of signal.

If the resources cannot be increased, there is no other possibility but decrease the displacement factor so that the next movement (that would cause the loss of signal in some room) doesn't have to be performed at all.

The Prolog code needs to be modified as follows:

Procedure: usage compensation with additional check

```
1: map$_1$(node,position).
2: map$_2$(node,position).

3: check_jaccard(node,$\delta$):-
4:      J(map$_1$(node,p$_1$),map$_2$(node,p$_2$)>=$\delta$/100,!.

5: check_jaccard(node,$\delta$):-
6:      J(map$_1$(node,p$_1$),map$_2$(node,p$_2$))<$\delta$/100,
7:      try_movement(m,node,map$_1$),empty_areas(m),
8:      decrease$\delta$_increaseResources(map$_1$),!.

9: : check_jaccard(node,$\delta$):-
10:      J(map$_1$(node,p$_1$),map$_2$(node,p$_2$))<$\delta$/100,
11:      try_movement(m,node,map$_1$),!empty_areas(m),
12:      move(node,map$_1$),check_jaccard(node,$\delta$).
```

As shown in the code, an additional check needs to be performed: the movement of the node in the map is tried before being performed. If projecting the current movement some areas end up being uncovered, the movement is not performed and the procedure to decrease the displacement factor or increase the number of available hubs is executed. Otherwise, the movement is effectively performed, and the procedure is recursively called.

V. CONCLUSIONS

One of the most important issues when one deals with wireless sensor networks is localization: not only it is crucial to locate objects all over the network, but it is essential to understand where hub should be placed in order to guarantee that some metrics are respected and exploited. Clearly, many proposals have been made in literature concerning the localization topic but, in this and our previous work, we use a logic approach based on Prolog facts and rules to simulate the hubs positioning over the interesting environment, based on successive approximation of nodes distribution: the relative position of each node is opportunely arranged at each iteration in such a way that the metric taken into account is respected.

While in our previous work we focus on the coverage property, meaning that the hubs distribution was only aimed to optimize the signal spreading in each area of the reference environment, in this work we improve this approach by adding another meaningful metric: the usage of the network.

Intuitively, the resources supply can be better performed if we focus on the areas where the usage of the network is very high, rather than areas where no one uses the provided signal.

From the other hand, a displacement in favour of usage statistics can seriously impact the coverage property, meaning that with this approach whole rooms could be not covered at all by any kind of signal. This is obviously something to avoid: for this reason, we introduce a displacement factor, which represents how much one is ready to sacrifice the coverage in favor of a better user-based distribution of the signal. We have shown some Prolog code lines that explain how this strategy works: two maps are used, the first one is the one resulting from our previous approach, and the second one is the one with a hundred percent of displacement factor. None of them constitute the final solution, which is rather a trade-off between them: indeed, by using the Jaccard similarity measure we managed to have a final map with a minimum displacement factor required, but without losing the coverage metric.

We went through two approximations of our solution: first, we have considered the possibility to move the nodes in the first map according to the displacement factor which determines the minimum similarity required for each pair of corresponding nodes in the two maps. This solution clearly solves the problem of introducing the user's information in the node distribution, but still does not guarantee that the coverage property is kept: this is the reason why the second approximation has been done, by introducing an additional check, ensuring that no areas can became free of signal at all.

References

[1] Lewis, F. L. (2004). “Wireless sensor networks”. Smart environments: technologies, protocols, and applications, 11, 46.

[2] Kumar, A., Shwe, H. Y., Wong, K. J., & Chong, P. H. (2017). Location-Based Routing Protocols for Wireless Sensor Networks: A Survey. *Wireless Sensor Network*, *9*(01), 25.

[3] Ssu, K. F., Ou, C. H., & Jiau, H. C. (2005). Localization with mobile anchor points in wireless sensor networks. *IEEE transactions on Vehicular Technology*, *54*(3), 1187-1197.

[4] Mao, G., Fidan, B., & Anderson, B. D. (2007). Wireless sensor network localization techniques. *Computer networks*, *51*(10), 2529-2553.

[5] Chintalapudi, K., Padmanabha Iyer, A., & Padmanabhan, V. N. (2010, September). Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking* (pp. 173-184). ACM.

[6] Parker, R., & Valaee, S. (2007). Vehicular node localization using received-signal-strength indicator. *IEEE Transactions on Vehicular Technology*, *56*(6), 3371-3380.

[7] Liu, C., Wu, K., & He, T. (2004, October). Sensor localization with ring overlapping based on comparison of received signal strength indicator. In *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on* (pp. 516-518). IEEE.

[8] Rong, P., & Sichitiu, M. L. (2006, September). Angle of arrival localization for wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on* (Vol. 1, pp. 374-382). IEEE.

[9] Severino, R., & Alves, M. (2007, June). Engineering a search and rescue application with a wireless sensor network-based localization mechanism. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a* (pp. 1-4). IEEE.

[10] Savvides, A., Park, H., & Srivastava, M. B. (2002, September). The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications* (pp. 112-121). ACM.

[11] Belloni, F., Ranki, V., Kainulainen, A., & Richter, A. (2009, March). Angle-based indoor positioning system for open indoor environments. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on* (pp. 261-265). IEEE.

[12] Priyantha, N. B., Balakrishnan, H., Demaine, E., & Teller, S. (2003, November). Anchor-free distributed localization in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems* (pp. 340-341). ACM.

[13] Mourad, F., Snoussi, H., Abdallah, F., & Richard, C. (2009). Anchor-based localization via interval analysis for mobile ad-hoc sensor networks. *IEEE Transactions on Signal Processing*, *57*(8), 3226-3239.

[14] Balzano, W., Del Sorbo, M. R., Murano, A., & Stranieri, S. (2016, November). A logic-based clustering approach for cooperative traffic control systems. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (pp. 737-746). Springer, Cham.

[15] Balzano, W., Del Sorbo, M. R., & Stranieri, S. (2016, March). A logic framework for c2c network management. In *Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on* (pp. 52-57). IEEE.

[16] Balzano, W., Murano, A., & Stranieri, S. (2017). Logic-based clustering approach for management and improvement of VANETs. *Journal of High Speed Networks*, *23*(3), 225-236.

[17] Balzano, W., & Stranieri, S. (2017, November). LoDGP: A Framework for Support Traffic Information Systems Based on Logic Paradigm. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (pp. 700-708). Springer, Cham.

[18] Giordano, M., Polese, G., Scanniello, G., Tortora, G. (2010, February). A System for Visual Role-Based Policy Modelling: In *International Journal of Visual Languages and Computing*, Vol. 21, No. 1, Elsevier, pp. 41-64.

[19] Balzano, W., Murano, A., & Vitale, F. (2016, March). WiFACT--Wireless Fingerprinting Automated Continuous Training. In *Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on* (pp. 75-80). IEEE.

[20] Balzano, W., Murano, A., & Vitale, F. (2016). V2V-en–vehicle-2-vehicle elastic network. *Procedia Computer Science*, *98*, 497-502.

[21] Balzano, W., Murano, A., & Vitale, F. (2018). SNOT-WiFi: Sensor network-optimized training for wireless fingerprinting. *Journal of High Speed Networks*, *24*(1), 79-87.

[22] Balzano, W. Stranieri, S. (2018). A Logic Range-Free Algorithm for Localization in Wireless Sensor Networks, in *The 24th International DMS Conference on Visualization and Visual Languages*, San Francisco, California, 2018. DOI reference number: 10.18293/DMSVIVA2018-008.

Smart City Control Room Dashboards: Big Data Infrastructure, from data to decision support

P. Bellini, D. Cenni, M. Marazzini, N. Mitolo, P. Nesi, M. Paolucci
DISIT lab (http://www.disit.org http://www.km4city.org)
University of Florence, pierfrancesco.bellini@unifi.it, daniele.cenni@unifi.it, mino.marazzini@unifi.it, paolo.nesi@unifi.it, michela.paolucci@unifi.it

Abstract: Smart City Control Rooms are mainly focused on Dashboards which are in turn created by using the so-called Dashboard Builders tools or generated custom. For a city the production of Dashboards is not something that is performed once forever, and it is a continuous working task for improving city monitoring, to follow extraordinary events and/or activities, to monitor critical conditions and cases. Thus, relevant complexities are due to the data aggregation architecture and to the identification of modalities to present data and their identification, prediction, etc., to arrive at producing high level representations that can be used by decision makers. In this paper, the architecture of a Dashboard Builder for creating Smart City Control Rooms is presented. As a validation and test, it has been adopted for generating the dashboards in Florence city and other cities in Tuscany area. The solution proposed has been developed in the context of REPLICATE H2020 European Commission Flagship project on Smart City and Communities.

Keywords: smart city dashboard, decision support system, widget, control room.

1. Introduction

In the development of a Smart City there is a great emphasis to the set-up of the so-called Smart City Control Room, SCCR. A SCCR is an area in which all the data are collected, aggregated and where high-level data/results are summarized and made accessible for the decision makers and shared to the city operators. In large metropolitan cities, the SCCR includes large panels/monitors (even covering large walls) in which the status of the city is reported in real-time presenting the view of the city with some synthesis, predictions, alert of data regarding: mobility, energy, social activities, environment, weather, public transportation, people flow, health, water, security, ICT, governmental, first aid, civil protection, police (118/112/911), fire brigade, hospital triage, and thus almost all the city resources expressed via KPI (Key Performance Indicator). Most of the KPI are representative of the status of resources deployed in the city and may be not geo-localized. Some of the city monitored resources are representative of the critical infrastructures for the city functionality and for the life of city users such as: transportation, energy, security, health, water, civil protection, ICT, etc. In medium sized cities, the daily management of city resources is performed by a set of *city operators*, which could be legally independent with respect to the central municipality. Thus, they autonomously manage their control rooms, accessing and rendering their own data to take their own decisions which may be limited in scope, and according to some defined protocols and strategies, [1], [2]. For example, when the energy network has a problem in an area of the city the energy is rerouted to reach the all possible subareas via a different path; when the water service network has a problem on major distribution tubes, the water is provided in other means (for example using tanks); in presence of traffic congestion the red-light timing is acted to facilitate the flow and bus paths are changed/rescheduled according to planned schemas.

Once identified and understood the needs of having an integrated SCCR, it is necessary: (i) to choose what must be shown on the panels on walls; (ii) to choose what must be shown on computers of the operators in the rooms and remotely connected; (iii) to understand how the data have to be collected and computed (in the case of prediction and early warning). The ingestion, aggregation and data analytics processes are very complex to be managed since the information is heterogeneous (different format, providers, protocols, etc.) and the total amount of data is a Big Data problem, moreover, the final indicators, provisions and suggestions calculated by these complex processes must be easily understood by the observers of the panels. It is a problem of data representation which also must consider the competence of the final users: citizens /observers/operators/experts/decision makers/etc. In most cases, the final users have to be trained to understand the data and graphics representations. They must become confident on what they see to understand in deep all the single details represented on the screen. They are not going to have time to learn when a critical event happens. Some data or events are easy to be represented and understood in a dashboard, for example: a traffic representation observing the city map with red, orange, yellow, and green segments on the streets; a sensors measuring some values like temperature, humidity percentage, etc.; while it is more difficult to comprehend other, more complex, kinds of

DOI reference number 10.18293/VLSS2018-030

data/results/provisions, for example: tables of pollution and pollination; traffic flow trends by numbers, etc. To make it easy to read even the most complex metrics, it is possible to use: alarm signals in red, blinking signal, etc. [2].

From the technical point of view, the tools for rendering information on SCCR are typically called Dashboards and are supported by Big Data aggregation tools [4]. The Dashboards should be capable to present real time data in several different manners with real time updates on screen autonomously H24 7/7 days, according to the refresh time of each data source, and have to be interactive to allow the users to make drill down activities on data to better understand the situation and the context. Dashboards for control rooms should not be confused with business intelligence tools that produce graphics from the combination of data extracted from some sources (database, files, API, etc.). In most cases, business intelligent tools may access to data with faceted indexing and search, for example in SOLR or ElasticSearch, [24], [25] . Those kinds of Dashboards are focused on single view of data, filtering and drilling down on data, rather that representing the city KPI and status. Examples of tools for drilling down on time and facet can be obtained by using Apache Banana ([26]), or HUE ([27]).

Moreover, the concept of Dashboard for SCCR is also often confused with the data aggregator tool that is a fundamental tool for the Control Room and city control in general and can be regarded as the back office of the Control Room. Many solutions for control rooms and their back-offices has been proposed such as those of IBM [5] on services for citizens, business, transport, communication, water and energy; [6] on governmental, educational, e-health, safety, energy, transport and utilities; etc. Most of these solutions present a multi-tier architecture ranging from 3 to 6 layers [7], [8], [9].

In this document, the Dashboard solution developed in the context of REPLICATE research and development projects of the European Commission is presented. Replicate is an SCC1 project of the European Commission on H2020 ([10]). The solution proposed is been based on Km4City Smart City Ontology ([11]) and data aggregator [4], [10], [22]. Please note that the Dashboard Management System of DISIT Lab is released as Affero GPL Open Source on GitHub, see DISIT lab page. The present solution is managing more than 1.2 million of complex events/data per day.

The paper is structured as follows. In Section 2, the requirements of smart city control room are discussed. Section 3 presents the adopted smart city architecture. In Section 4, the dashboard system for the smart city control room is presented with its architecture. Section 5 reports a set of experimental results and lesson learnt. Conclusions are drawn in Section 6.

2. Smart City Control Room Requirements

In this section, the main requirements of Dashboard systems for SCCRs are summarized. They have been collected during the above-mentioned research project by interviewing a number of operators and decisions makers belonging to several cities and nationalities.

A SCCR dashboard system is substantially a Decision Support System tool, DSS, since it provides evidence of critical conditions, and may offer solutions. On this regard, it may integrate/exploit artificial intelligence algorithms, for example, reporting prediction, identifying anomalies, manifesting early warning, providing relationships among entities exploiting inference geospatial reasoning about what is located in the city: resources, structure, people, areas, critical infrastructures, etc. [10], [17], [19], [20].

According to our analysis, a Dashboard system for smart city must be capable to:

- Show dashboard on web browser in a H24/7 modality;
- show data on widgets according to several graphic paradigms (tables, graphs, histograms, maps, Kiviat, lists, tv camera, heatmaps, weather, critical city events, etc.) with a level of interactivity and animation;
- show data on autonomous and connected/ synchronized widgets;
- collect, show and keep update on screen data with automated refresh for each view, and real time data according to the even driven paradigm;
- show data both real time and historical, allowing the drill down on time, space and relationships among data and city entities;
- collect and show data coming from different big data and classic data sources (SQL, NoSQL, RDF, P2P, API, SOLR, etc.) also in aggregated manner;
- compose the Dashboards as a set of graphic and integrated widgets that can be separately set up assigning several parameters: data source, size, colors, shape, etc.;
- work with large amount of data providing high performances, as short response time;
- compute alarms, and provide support by a flexible notification system capable to send alerts, activate tickets for maintenance, automate actuators, post on social, etc.;
- provide actuators widgets together with showing graphs, and capable to act on IOT Devices;
- provide support for collaborative production of dashboards and for co-working;
- provide support for embedding dashboards into third party web pages;
- provide data engine for collecting connection response time on different protocols, and for verifying the consistency of web pages via HTTP;
- integrating with IOT Applications by managing real time data and connecting its actuators to real time IOT applications;

- integrate dashboards in more complex dashboard systems;
- support authentication and authorization with the most general approaches such as LDAP, Kerberos, etc.;
- collect and get data from batch resources and in real time, using a large range of protocols and formats.

This means that each Dashboard should be composed by several configurable Widgets, each of them can collect data coming from several data sources. On each data stream, one or more criteria as firing conditions should be set up for the notification of alerts, intervention, etc.

In small cities, with about 100.000 inhabitants the number of relevant data sources to be integrated by the data aggregator and represented in Dashboard can be in order of 20-30 while in larger cities they can rapidly grow for the presence of multiple operators for each utility. Thus, the complexity is also greater as the number of stakeholder actors involved in.

Therefore, before starting with the development of the proposed Dashboard solution, several state-of-the-art solutions and proposals have been analyzed. As nonfunctional requirements, the Dashboard system must be scalable, interoperable with several tools, open source, robust, usable, secure in protecting data views according to the GDPR [17] and flexible.

A set of solutions, both commercial and open-source, have been analyzed to identify a functional platform to be adopted. Most of the solutions which are present at the state of the art derive from business intelligence solutions (e.g., SpagoBI, Tableux, OpenDataSoft, etc. [13], [14], [16]), in which the tools provide some data mart (data virtualization) tool to access data sources and thus have powerful tools in this sense. On the contrary, they provide limited capabilities and tools on rendering and dashboard for control rooms that must stay H24/7, rendering specific kind of structured data. For these reasons, several specific custom solutions have been proposed by many cities such as: London (https://data.london.gov.uk/), Amsterdam (http://citydashboard.waag.org/), Dublin, etc. On the bases of the analysis made, regarding the solutions available on the market, none of them satisfied all the requested functionalities and functional aspects, above described, and this is the reason why we decided to start the development of our solution, ([14]).

3. Smart City Architecture

This section presents the overview of the Smart City architecture which is presently in place in the Tuscany area. With the aim of producing a smart city infrastructure for stimulating sustainable mobility, smart energy, and smart utility in the city, a data aggregator has been developed (see **Figure** 1). It presents a front-end layer for the City Dashboard and control room, Smart City API for web and mobile App, decision support tools, personal assistants via mobile App, participative portals, crowd sourcing, etc. The data aggregation infrastructure also supports Data Analytics and Data Intelligence based on integrated data collected from public administrations open data, private data from operators, and personal data coming from social media and city users. In this paper, the architecture enabling the construction of the Control Room in terms of Dashboards and Data Aggregator is described.

In the Km4City layer, **City Operators and Data Brokers** provide data which are collected by using streams, data driven and/or ETL processes which are scheduled on the Big Data processing back office based on DISCES (Distributed Smart City Engine Scheduler) tool. Among the data collected those provided in Open Data from the municipalities, Tuscany region (Observatory of mobility), LAMMA weather agency, ARPAT environmental agency, etc., and several private data coming from City/Regionals Operators: mobility, energy, health, cultural heritage,

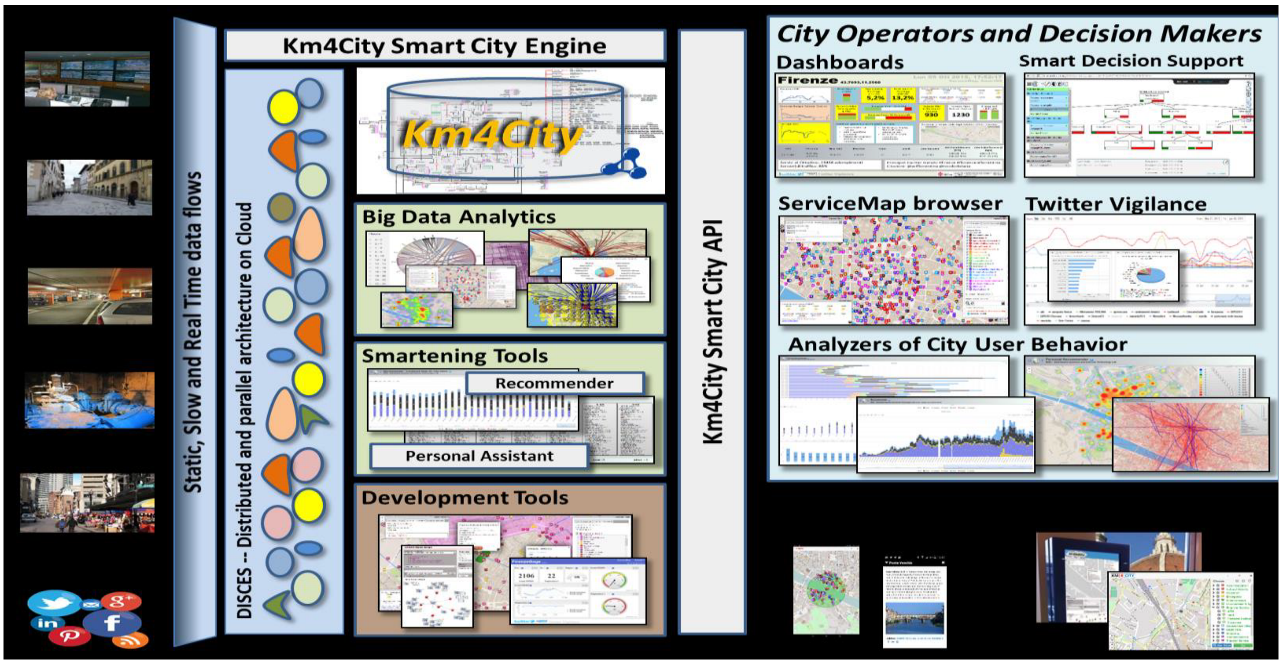

Static, Slow and Real Time data flows
Km4City Smart City Engine
DISCES -- Distributed and parallel architecture on Cloud
Km4City
Big Data Analytics
Smartening Tools
Recommender
Personal Assistant
Development Tools
Km4City Smart City API
City Operators and Decision Makers
Dashboards
Smart Decision Support
ServiceMap browser
Twitter Vigilance
Analyzers of City User Behavior

Figure 1: Km4City Architecture for Smart City.

services, tourism, wine and food services, education, wellness, environment, civil protection, weather forecast, etc. Once the data are collected, the back office activates several processes for improving data quality, reconciliating data and converting data into triples for the RDF store of the Knowledge Base (when needed) [10], [4], implemented by using a Virtuoso triple store. DISCES is allocating processes on several virtual machines allocated on the cloud according to their schedule.

For semantic aggregation of data and service, it has been decided to exploit and improve the Km4City Ontology ([10]) as the main ontological model. Km4City is modeling multiple domain aspects related to mobility, services, Wi-Fi, cultural services, energy, structure (streets, civic numbers, public structures, geographic locations, green areas), sensors, busses, smart sensors, parking, city services, transportation, events, pharmacies, hospital and real time data of first aid, environment (with pollution and pollination, weather forecast, weather conditions), and private mobility with fuel prices.

In the smart city architecture, in addition to the RDF store for the knowledge base, a number of noSQL Stores (namely: HBase and MongoDB, [28], [29]) are adopted for storing tabular data as those arriving from sensors and user profiles, respectively.

4. Dashboard System Architecture

In **Figure 2**, the general architecture of the Dashboard solution is presented. The main components of the architecture are described in the following.

- **Data Aggregator** is a set of tools for collecting data from the field and from external services and reconciling them to the same city entities. To this end in the proposed architecture the Km4City Aggregation has been adopted to have all data fit into the Km4City Ontological model Thus. Data are provided from the Aggregator with Smart City API (rest CALL), SPARQL, SOLR and/or SQL queries.
- **Data Collector** is a multi-process engine (also called Dashboard Engine) for acquiring data from multiple data sources: SQL, noSQL, RDF/SPARQL, API, SOLR, etc. ([25],[28],[29],[30],[31]), by using multiple protocols: HTTP, HTTPS, ODBC, WebSocket, etc. The Data Collector needs to have a configuration for each acquisition process, which produce a result, also named **Metric** or Measure. Some of these **Metrics** may be saved into a local data base for historical reason. To finalize the queries to be performed for collecting Metrics, specific tools may be used for **Data Mart** such as database browsers and drilling down into Data Sources. The collected data can be (i) saved into a data base of **Metric Historical Values** or can be (ii) directly accessed from Widget/Dashboard for their visualization. The Data Collector, with its processes to acquire Metrics, is also capable to perform the real time assessment of firing conditions.
- **Firing Assessment** allows to compute the firing conditions on all the data streams reaching the dashboard system. In the case in which, a Firing Condition becomes true, a message event is sent to the Notificator service. Firing Conditions can be estimated:

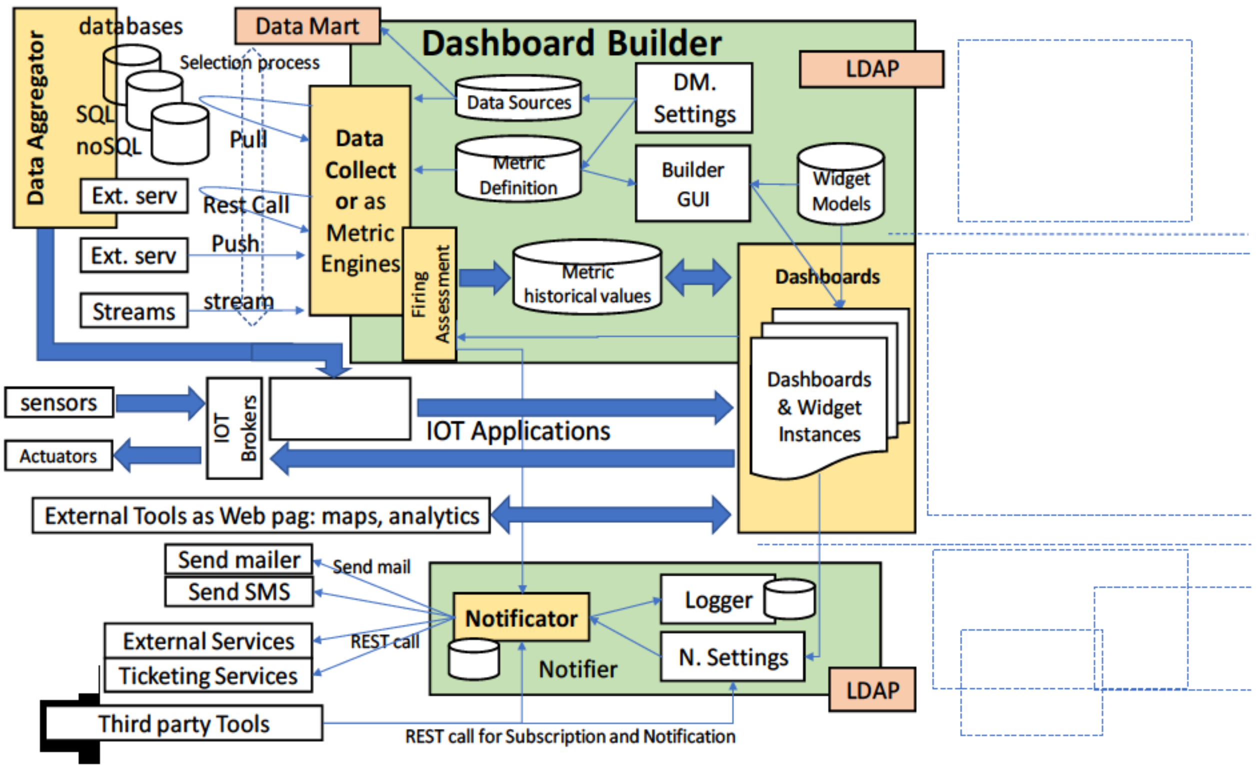


Figure 2: Architecture for Smart City Dashboard Management System integrated with IOT.

    - on Metric Historical Values taken in Push from the data base of the builder;
    - on the streams directly arriving from the Data Collector;
    - from the data stream arriving directly from outside,
    - From the external tools embedded as IFRAME into widget if there is some integration.
- **Dashboard Builder** is the core tool for creating dashboards by using a graphic user interface. In the tool the user can set up **Data Sources**: IP address, protocol, user name and password for accessing at each specific the data source. Once the data sources are identified several **Metrics** can be defined. Then new Dashboards can be created taking interested **Metrics** and associating them with one or more **Widget.** A Widget may render/exploit the same Metric by means of different graphic models. For example, a temperature read every 5 minutes, can be visualized as current value on a thermometer, as temperature trend in a graphic along the last 24 hours, last week, last month, etc. Therefore, the composition of a **Dashboard consists of** placing and configuring a set of Widgets into the Dashboard frame. The Dashboards are created by using the visual interface of the **Dashboard Builder**.
- **Integration with IOT.** This feature has been covered by (i) producing special MicroServices as blocks that can be used into IOT applications developed in NodeRED, (ii) connecting NodeRED applications with several IOT brokers, ([33]). Point (i) implied the development of a layer that allowed the traditional Dashboard Widget to be directly connected to (a) IOT Applications by using WebSockets, (b) IOT Brokers (for example, via NGSI, MQTT, etc.). To this end, one of the most suitable IOT Brokers resulted to be the Fi-Ware Orion Broker.

Dashboards are typically adopted for reporting KPI of the city and thus of specific infrastructures and services. This means that specific alerts and notifications have to be activated and managed at level of single Metric. On the other hand, the same Metric can be used on different dashboards and widgets for different purposes. So that, for each Widget of each Dashboard specific alerts and firing conditions can be set up. For example, when Metric M is adopted in Widget W of Dashboard D, the certain criterion C is saved and computed for firing (M, W, D, C). One or more Criteria can be defined (M, W, D, C1 … Cc), each of them may produce multiple Notifications,
N: (M, W, D, C1 (N1, …, Nn), …, Cc (…)).

Therefore, the solution has been to design and develop a **Notifier** to

- Accept registrations of possible tuples <m,w,d,c>, to enable the reception of *Notification Requests*, that are used to send Notifications according to different approaches;
- Accept registration by third-party tools, in addition to those of the Dashboard Builder, to send alarms about the firing of the registered conditions;
- Produce emails and REST calls, that can be used for calling SMS, as well as for the activation of maintenance ticketing system on OpenMaint tool for example ([32]);
- Log all the registrations and Notification Requests for further analysis and security evidence;
- Maintain and use a list of Notification recipients, that are the users which are going to receive the notifications. This list of uses is just listed as: name, surname, email, telephone (if any), role, organization.

To this end we suggest using specifically development tools, Such as the ServiceMap ([34]) which is used for knowledge base browsing over the city data as Km4City knowledge base, which is RDF store as well, exploiting geospatial reasoning and inference [Bellini et al., 2014]. In addition, the technical browsing on the RDF Graph Store may be needed to discover relationships. To this end, the LOG ([35]) tool for browsing into any RDF store by using SPARQL and visual interface has been developed in the past and now used in this context. This tool allows to browse all the RDF stores accessible in the world which provide a public end point for SPARQL queries, from dbPedia, to Europeana, Geonames, Km4City, Camera, Senato, Getty, etc. [23].

As a conclusion, the generated Dashboards are **Dashboard Instances** which are available for view and activate corresponding widgets according to their Settings. The saving of data into the database of **Metric Historical Values**, allows keeping track of what has been visualized/monitored and thus enabling the replay of data logged. On the other hand, it is also possible to adopt widgets that: (i) directly show/provide the data from in/out streams, respectively (for example, Civil protection alert status, etc.); (ii) directly render/visualize web page segments into the Dashboard (for example for showing social media analytics, traffic flow reconstruction tools).

5. Experimental Results and Validation

The solution proposed in this paper has been developed in REPLICATE flagship research and development project SCC1 H2020 of European Commission for Florence City. It has been also used in other large projects such as Sii-Mobility Smart City Nazionale on Mobility and Transport of MIUR, RESOLUTE H2020 project for critical infrastructure and resilience of transport systems, and GHOST MIUR for Cagliari smart city experimentation. The proposed Dashboard solution is strictly connected with a number of tools of the Sii-Mobility/Km4City suite of tools which are used to perform smart city analytics, semantic browsing, and decision support, etc., such as: ServiceMap ([34]), smartDS, system thinking decision support based on Bayesian models (http://smartds.disit.org [21]), Wi-Fi monitoring and predictions, smart parking prediction, traffic flow reconstruction and prediction, energy metering, first aid monitoring, environmental monitoring, social media monitoring and alerting, weather forecast, etc.; most of them based on clustering, machine learning, etc. [4].

In general, the decision makers in the city are politicians, assessors, and director of units. Some of the units have already adopted a high level of technology, for example, the mobility and transport, the civil protection, etc. In other units, the level of control is low since the technical activity is mainly delegated to City Operators such as: energy operators, water management, health care hospitals, environmental agency, waste management, police and alert (112, 118, 911), etc. All of them have their own monitoring system, that is tuned to operators. The dashboard can organize data according to different views/paradigms: horizontal view (synthetic view reporting many different aspects of the city status) and vertical (or thematic).

5.1 Examples of Horizontal User Oriented

A horizontal dashboard contains a synthetic view reporting many different aspects of the city status, such as:

Event oriented: a dashboard for controlling the status of city with respect to a large event (such as the visit of Pope, or of the US President). In that case, the dashboard would be dedicated to monitor the paths that would be probably used to reach some point of interest, the status of traffic in those area and the major points that may influence that area (directly and indirectly), the number of police and security resources in those area, the TV cameras, the hospitals, the aggregation area, the other events and micro-events (accidents, crashes, fights), etc.

Tourism oriented: a dashboard reporting the major

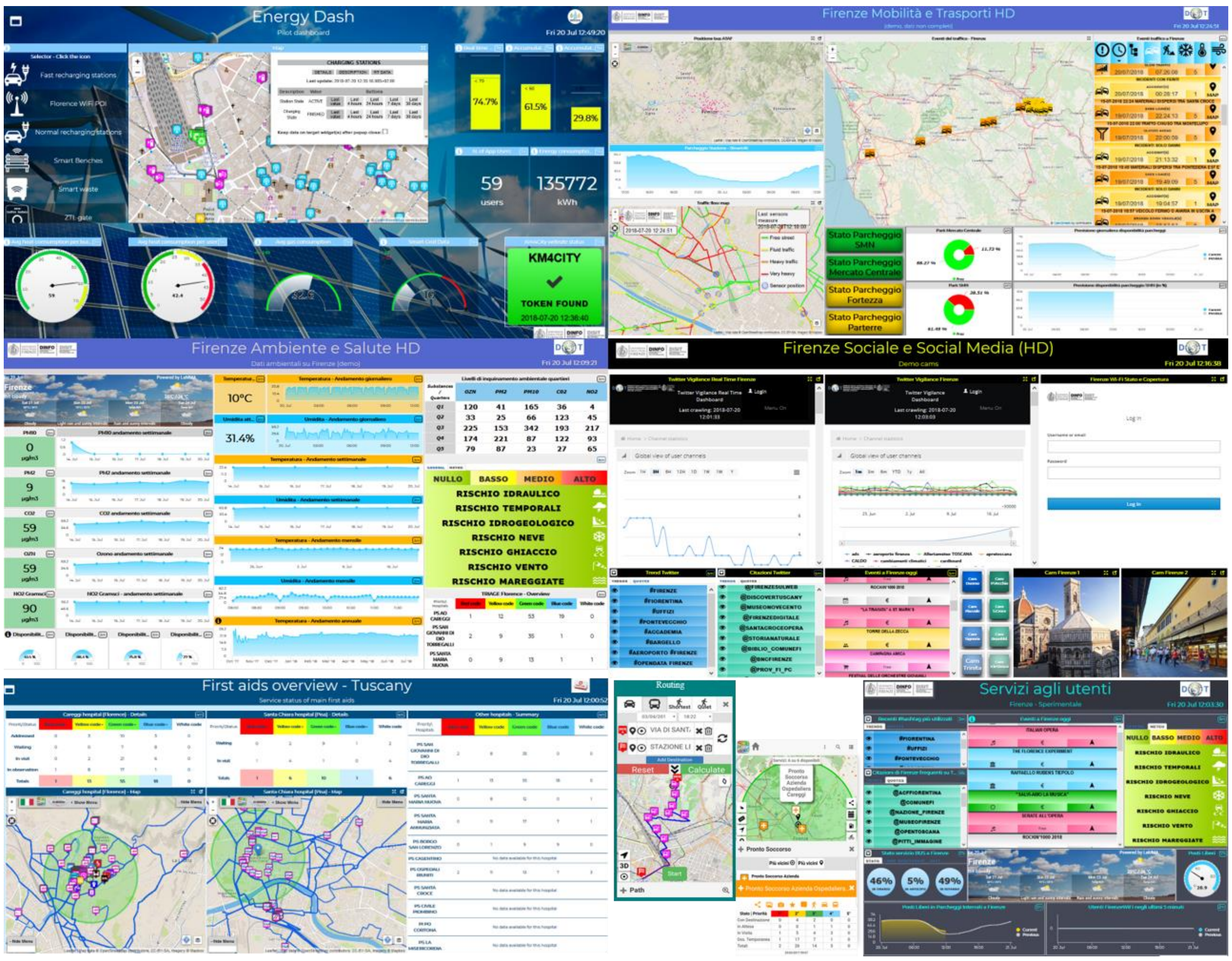

Energy Dash
Firenze Mobilità e Trasporti HD
Firenze Ambiente e Salute HD
Firenze Sociale e Social Media (HD)
First aids overview - Tuscany
Servizi agli utenti
KM4CITY
TOKEN FOUND
59 users
135772 kWh

Figure 3: Florence Smart City Dashboards, dashboard reporting first aid status, and a final user dashboard for hotels.

vertical control their own information and status. In some cases, the general information about weather forecasts and status is shared among the different events in the city, the number of arrivals in the city, the number of people in the major points of the city, the number of accesses to the museums, the number of touristic busses arriving in the city and their paths, etc.

5.2 Example of Vertical thematic oriented

A thematic dashboard contains the available data related to a specific context. It can be viewed as an in-deep representation of one of the aspects reported in the horizontal dashboards. Some samples can be:

public transportation: position of busses in real time, number of active busses, average delay at the bus stops, number of active taxis with respect to the plan, number of recharging stations for public vehicles and their status, number of people on busses, percentage of busses with respect to the plan, number of events/incidents on traffic, status of the underpasses, status of the bridges, number of tickets, number of free lots in parking, events in the city, etc.

private mobility: level of traffic flow, traffic flow reconstruction with classic colors (green, yellow, orange and red), number of free slots in parking, number of cycling people on paths, number of vehicles entered into the RTZ per hour, number of vehicles entered in the city per hours, number of trucks on the main streets, number of shared bikes in percentage respect to the total available, events in the city, etc.

Energy: KW/h or GW/h consumed in the last hour for public services, KW/h or MW/h consumed in the last hour for recharge stations, KW/h or MW/h saved by public services since the usage of renewable energy production, KW/h or MW/h saved in the store and available for consumption, number of monitored meters grid, saved energy by following suggestion provided by Apps, Co2 saved by using e-Vehicle, etc.

Environmental data referring to different area of the city: temperature, humidity, PM10, PM2, CO2, wind, pollination, etc.; weather forecast; real time data from environment (temperature, humidity, wind velocity, etc.); level of water in the rivers; level of drinkable water; Tons of collected garbage; Tons of collected garbage differentiated; earthquake monitoring; etc.

Social: status and stream of the social media; the most cited users on Tweet; the most mentioned hashtags on Twitter.com; the sentiment analysis on Tweets connected to the city in the last minutes; number of people moving the main area; number of people arrived by train in the City; TV cameras about the main points of interest in the city; number and list of the major entertainments, political, and sport events in the city; etc.

Security: data also presented on the Social Dashboard describing the presences in the city of people; any kind of event in the city from entertainment, sport, political, religious, critical events on the road, etc.; eventual paths and area of the events; TV cameras observing the areas of the events; number of resources available for controlling the city and their deploy on the city map (cop, ambulances, 118, 911); aspects related to the environmental data; aspects related to mobility for planning the evacuations; aspects related to the public transportation for eventual change the paths.

Health: data reporting the status of the triage in the major hospitals; position of the emergency stations; number and position of the ambulances; environmental data for hot waves, temperature, etc., which can increase the risk of stroke.

5.3 Validation

As a conclusion, after the production of a set of Dashboards some of them where selected for trial and are reported in the Km4city portal, [11], where most of them are presenting public data that can be rendered on screen. This does not mean that the published data are open and that can be downloaded to be reused and published/exploited for other purpose. Moreover, the Dashboard can also contain data that cannot be visualized by public, for safety reasons, and thus are protected by some conditional access solution. For instance, since they are sensitive data describing the city status in real time or by predictions. Many examples of dashboards produced by the presented tool can be accessed from the Km4City Portal, [11].

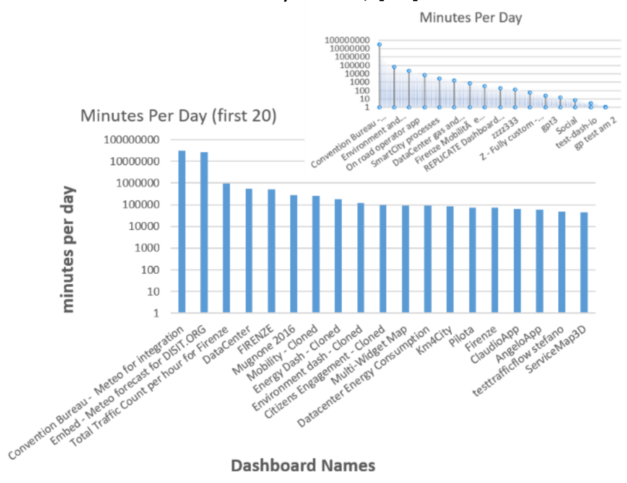


Figure 4: Minutes per Day. All dashboards and only first 20.

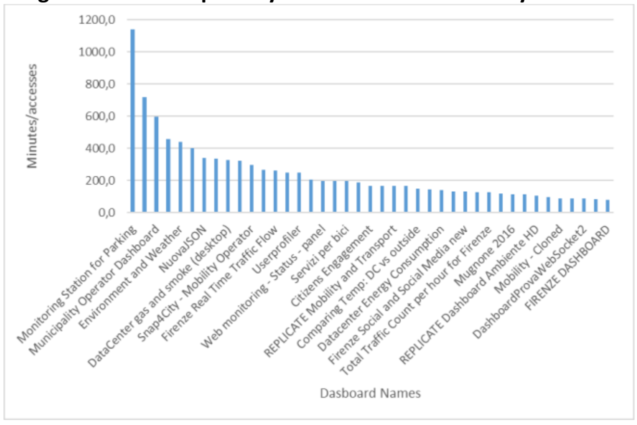


Figure 5: Average number of minutes the people stay connected on dashboard for each access

Actually, 653 Dashboards are present in the system: the total number of minutes of access to the dashboards is 62.410.108 (29.045,28333 total hours) and the total number of clicks on them is 1.742.798, 33% are public and the 67% are private. Private dashboards are those accessible only by their author while the others are visible to all the web users. In Fig.4 it can be seen how many minutes per day each dashboard is accessed by the users: at the top right you can see the trend of all the dashboards, at the bottom left only the dashboards that have a total greater than minutes of access. In Fig.5, it is possible to see the average number of minutes the people stay connected on dashboard for each access (calculated as {total minutes per day}/{total accesses per day). This second instagram, is relevant to evaluate which type of dashboard is best suited to stay in a control room means: the dashboards having the highest values are those kept longer on the screen and more comfortable as staying in a control room.

6. Conclusions

Smart cities are becoming more and more advanced, for this reason it is necessary to ingestion a multitude of data not only static and periodic but in real time. This amount of data (Big Data) must be analyzed to provide information on the state of the city both to citizens and especially to decision makers. A fundamental aspect is to study and apply ad hoc methodologies to visualize the events in real time, in Smart City Control Room Dashboards. In this work we have analyzed the functional characteristics that these Dashboards must have to better represent the state of the city. The analysis has led us to conclude that there are no solutions on the market that meet all the requirements outlined, so it has led us to develop our own solution which fits the Km4City Big Data architecture.

7. Acknowledgements

Thanks to the European Commission for founding. All slides reporting logo of REPLICATE H2020 are representing tools and research founded by European Commission for the REPLICATE project. REPLICATE has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement n° 691735).

8. References

[1] M. Azzari, C. Garau, P. Nesi, M. Paolucci, P. Zamperlin, "Smart City Governance Strategies to better move towards a Smart Urbanism", The 18th International Conference on Computational Science and Its Applications (ICCSA 2018), July 2 - 5, 2018 in Melbourne, Australia in collaboration with the Monash University, Australia.

[2] C. Garau, P. Zamperlin, M. Azzari, P. Nesi, G. Balletto, M. Paolucci, THE ROLE OF KM4CITY DASHBOARD IN URBAN POLICIES: GOVERNANCE STRATEGIES FOR DYNAMIC URBAN SYSTEMS from 2nd International Conference on Smart and Sustainable Planning for Cities and Regions 2017, Bolzano/Bozen (Italy), 22-24 March 2017.

[3] Few, Stephen. "Information dashboard design." (2006).

[4] C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi, M. Paolucci, Analysis and Assessment of a Knowledge Based Smart City Architecture Providing Service APIs, Future Generation Computer Systems, Elsevier, 2017.

[5] IBM Institute for Business Value, "How Smart is your city? Helping cities measure progress", [online]. Available: oct 2013, http://www.ibm.com/smarterplanet/global/files/uk__en_uk__cities __ibm_sp_pov_smartcity.pdf

[6] Alcatel-Lucent Market and Consumer Insight team, "Getting t about Smart Cities Understanding the market opportunity in the cities of tomorrow", Oct. 2013.

[7] Anthopoulos L., Fitsilis P. "Exploring architectural and organizational features in smart cities." Advanced Communication Technology (ICACT), 2014 16th Int. Conference on. IEEE, 2014.

[8] Filipponi L., Vitaletti A., Landi G., Memeo V., Laura G.; Pucci P., "Smart City: An Event Driven Architecture for Monitoring Public Spaces with Heterogeneous Sensors," in Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on , vol., no., pp.281-286, 18-25 July 2010.

[9] Chourabi, Hafedh, et al. "Understanding smart cities: An integrative framework." System Science (HICSS), 2012 45th Hawaii Int. Conference on. IEEE, 2012.

[10] Replicate Project, http://www.replicate-project.org

[11] Km4City, https://www.km4city.org

[12] P. Bellini, M. Benigni, R. Billero, P. Nesi and N. Rauch, "Km4City Ontology Building vs Data Harvesting and Cleaning for Smart-city Services", International Journal of Visual Language and Computing, Elsevier, 2014.

[13] SpagoBI, http://www.spagobi.org/

[14] P. Bellini, D. Cenni, M. Marazzini, N. Mitolo, P. Nesi, M. Paolucci, "Smart City Control Room Dashboards Exploiting Big Data Infrastructure", The 24th International DMS Conference on Visualization and Visual Languages, DMSVIVA 2018, Hotel Pullman, Redwood City, San Francisco Bay, California, USA, June 29 - 30, 2018.

[15] Tableau, https://www.tableau.com

[16] OpenDataSoft, https://www.opendatasoft.com

[17] GDPR, General Data Protection Regulation, https://www.eugdpr.org

[18] Suakanto, Sinung, Suhono H. Supangkat, and Roberd Saragih. "Smart city dashboard for integrating various data of sensor networks." ICT for Smart Society (ICISS), 2013 International Conference on. IEEE, 2013.

[19] McArdle, Gavin, and Rob Kitchin. "The Dublin Dashboard: Design and development of a real-time analytical urban dashboard." (2016): 19-25.

[20] De Marco, Alberto, Giulio Mangano, and Giovanni Zenezini. "Digital Dashboards for Smart City Governance: A Case Project to Develop an Urban Safety Indicator Model." Journal of Computer and Communications 3.5 (2015): 144-152.

[21] E. Bellini, P. Nesi, G. Pantaleo, A. Venturi, "Functional Resonance Analysis Method based Decision Support tool for Urban Transport System Resilience Management", IEEE Int. Smart Cities Conference (ISC2), 12-15 Sept. 2016, Italy.

[22] Bellini P., DiClaudio M., Nesi P., Rauch N., "Tassonomy and Review of Big Data Solutions Navigation", as Chapter 2 in "Big Data Computing", Ed. Rajendra Akerkar, Western Norway Research Institute, Norway, Chapman and Hall/CRC press, ISBN 978-1-46-657837-1, eBook: 978-1-46-657838-8, july 2013, pp.57-101, DOI: 10.1201/b16014-4.

[23] Bellini P., Nesi P., Venturi A., "Linked Open Graph: browsing multiple SPARQL entry points to build your own LOD views", http://log.disit.org International Journal of Visual Language and Computing, Elsevier, 2014.

[24] Elasticsearch, https://www.elastic.co/products/elasticsearch

[25] Apache Solr, http://lucene.apache.org/solr

[26] Apache Banana, https://github.com/Lucidworks/banana/

[27] HUE, http://gethue.com/overview

[28] HBase, Apache HBase, https://hbase.apache.org/

[29] MongoDB, https://www.mongodb.com

[30] RDF https://www.w3.org/RDF/

[31] SPARQL: https://www.w3.org/TR/rdf-sparql-query

[32] Openmaint: http://www.openmaint.org/en/project/how-it-works

[33] Node-Red, https://nodered.org

[34] ServiceMap, http://servicemap.disit.org

[35] LOG, http://log.disit.org