Traditional Approaches in Knowledge Representation[1]

MOHAMMED SAEED, Ali Amer
University of Pitești[2]
ali.amer81@gmail.com

Abstract

In this paper, both declarative and procedural methods will be presented. The declarative methods are represented by semantic networks, frames, structured objects, and predicate computation. All these are used to represent facts and assertions or processing rules. Procedural methods mainly include production rules and are used to define actions or processes.

Keywords: *conceptual graphs, semantic networks, CRC, CGIF, CLIF, KIF.*

ACM Classification: I.2.4

1. *Relationships, knowledge bases, queries*

The representation of knowledge through relationships can be seen as the transfer of human expertise to knowledge base processing systems. Starting from Motro and Yuan (1990), which analyze how a relational database can be enriched so that it can provide intelligent queries, we differentiate between data queries and queries (i.e. knowledge queries). Access to both data and knowledge should be achieved by a single instrument. Expression of concepts is done using the concept map and emerged in the field of education, specifically in the learning and evaluation activities, according to Novak and Canas (2006). Any conceptual map has the following features:

1. it expresses the components of any process / concept (sometimes called semantic units) and relationships between its components;
2. the key concept / process is at the center of attention;
3. sub-concepts / sub-processes are placed to highlight dependencies or constraints;
4. the number of placed entities belongs to interval 10-15;

[1]Citation: Mohammed Saeed A.A., *Traditional Approaches in Knowledge Representation*, "An. Univ. Spiru Haret, Ser. Mat.–Inform.", 11(2), pp. 5–16, 2015.

[2]Doctoral School of Informatics

5. any graphic representation uses circles (or rectangles) and arrows.

The conceptual map should not be confused with the Memory Map. Formally, the conceptual map is an oriented graph in which the nodes represent the concepts (important semantic units), and the arcs are ties of determination between them (from general to particular) that express the relationship between two concepts or nodes. Conceptual mapping can be visualized using the CMAP software application: http://cmap.ihmc.us/cmaptools/cmaptools-download/.

Conceptual maps benefit from various graphical representations: a) spider web (in the center is the central concept from which the rays link to the secondary concepts); b) the tree structure (the leaves indicate the primary concepts and the internal and root nodes the concepts obtained by various operations, resulting in a hierarchy represented in the form of an arbitrary tree - in case the hierarchy is made by pairs of concepts, then a binary tree is obtained); c) general conceptual maps with inputs and outputs.

There are two ways to look at the link between systems for knowledge representation and processing (hereinafter KRP systems) and Database Management Systems (DBMS), according to Borgida (2007). On the one hand, it is about applying database and data processing concepts stored in databases to KRP systems. Thus, a KRP system needs to have adequate (persistent) storage techniques and a demand optimization module (scalability). Additionally, concurrent access to data, as well as recovery from demage events, should be ensured. On the other hand, specific KRP ideas (eg. descriptive logic) can be applied to design traditional databases using well-formed entity-relationship diagrams. Some examples of useful relationships in the practice of using databases are:

Products (product_code, product_name, unit_type, price),

Vendors (vendor_code, vendor_name, vendor_locator, phone_upplier, vendor_burner),

Employee (person_code, name, last_name, gender, salary, project_code, group_code).

A possible SQL description of the employee relationship is:

CREATE TABLE **Employee** (
Person_code SMALLINT,
Name VARCHAR (20),
Last_name VARCHAR (25),
Gender CHAR (1),
Salary INTEGER,
Project_code VARCHAR(4),
Group_code SMALLINT)

Transforming the above representations into intelligent interrogation structures is possible by rethinking information as part of a predicate (according to first order logic). An enriched base will be obtained [Motro and Yuan (1990)]:

- A set of predicates, denoted by P, and for each predicate a lot of facts (such as those in the database tables) are available. Each predicate is defined "true" for the associated facts and is "false", otherwise.

- A set of predefined predicates.
- A set of predicates, denoted by S, with a number of rules available for each predicate.
- The pools P, R and S are disjoint two by two.
- The entire database can be thought of as a set of axioms, and the deduction rule is based on the resolution principle. Thus, we can consider an extension of the database concept (DB), which becomes the Knowledge Base (KDB).

The union of the sets P and R forms the extension of the knowledge base, and the set S, which allows intelligent interrogation by rules, is the intensional component. In the following example (adapted from [Motro and Yuan (1990)]), there are eight predicates that form the extensional part and three predicates (intensional part) that allow intelligent queries on the data collection:

Table 1: Enriched base.

The set P	Student(student_name, specialization, media) Professor (professor_name, department, phone) Course (course_title, credits) Enrolled (student_name, course_title) Holder (professor_name, course_title) Conditioned (course_title_1, course_title_2) Supported (professional_name, course_title, semester, qualification) Follow_and_Passed (student_name, course_title, semester, note)
The set R	$=, \neq, >, \geq, <, \leq$
The set S	meritous (X): = student (X, Y, Z) and (Z$\geq$10) Precede (X, Y): = conditioned (X, Y) Precede (X, Y): = conditioned (X, Z) and precede (Z, Y) Prep (X, Y):= meritous (X) and Follow_and_Passed (X, Y, Z, U), and (U $\geq$ 8) and Supported (V, Y, Z, W) and Holder(V, Y) Prep (X, Y): = meritous (X) and Follow_and_Passed (X, Y, Z, 10)

The student X can be a trainer for the course Y if he has completed the course Y obtaining the maximum grade (10) or has at least an average of 10, and at Y has at least scored 8 at the course taught by the lecturer Y in the current semester. We note that the above allows for the implementation in PROLOG of a part of a KRP system for the management of educational activities.

Forming a request (creating a query) related to the employee table can be done as follows:

```
SELECT person_code, name, last name, gender, salary, project_code
FROM Employee
WHERE group_code ="CS";
```

The formulation of a request for the list of trainers for the "Fundamentals of Programming" course can be done by the querry Prep (X, "Fundamentals of Programming").

To conclude, all relational KRP systems allow entities to be compared based on equivalent attributes to infer the validity of the tested relationship.

2. ***Object oriented KRP methods***

Object Oriented Thinking is the test stone of both object-oriented programming and design of object-oriented databases, and why not, of reasoning about sentences in which objects occur. An application of object oriented KRP systems in medicine has been reported by Ensing et al. (1994). In order to design and implement an object-oriented KRP system, the following four phases must be pursued: domain characterization, object-oriented modeling, formalization, and actual implementation. Once the problem has been well understood, object-oriented modeling can use the CRC method considering all Classes, Responsibilities, and Collaborations.

One CRC card is designed for each class and indicates the responsibilities of the class as well as the classes that comes into contact with. For example, to create an application that creates PDF and SVG documents and containing text, circles, triangles, and squares, we can design classes called Circle, Square, Triangle and Text Class, a class for making the pdf document, and a class for making the SVG document. Thus, the Figure class has the responsibility to manage a list of geometric shapes, and as collaborators are the classes: Circle, Triangle, Text and Square. The card associated with the PDFDoc class describes as a responsibility the saving of a figure in PDF format and specifies the collaborative classes: Figure, Circle, Square, Triangle, and Text.

Table 2: CRC model.

Name of class:	
Responsibilities:	Collaborations:

CRC cards were introduced by K. Beck and W. Cunningham to illustrate how object-oriented concepts can be taught and learned. The structure of a CRC card, as suggested above, should specify: Name of the class, Fields (data and methods) of the class, and Types of entities with which this class collaborates: Simple types of data, predefined classes, and project classes. CRC cards are portable, do not require the use of a computer (if they work with cards), they allow the experimentation of approaches (by using the gum and pencil or the Del/Ins option in the computerized version).

The most important approach applicable to current object oriented modeling is based on UML - Unified Modeling Language. In UML, all the things in the universe of discourse are modeled using classes. A class means the description of a set of objects that share the same attributes, operations, relationships, and the same meaning. Class diagrams specify the structure of the system, i.e. the set of classes in the system and the multitude of links between them. In UML, a class is represented graphically by a rectangle inside which the class name is written. Each class is characterized by a variety of operations (generic behavior described by code organized as functions or methods) and attributes (describing the state of an entity). Components of a class may be public (+), private (-), or protected (#). The code can be changed in classes derived from the original ones, it can be inherited or can be final (it is no longer allowed to rewrite the method in the derived classes).

Each class has two specific methods: the constructor (which is responsible for creating an object and determining the initial state of the object) and the destructor (responsible for saving the context and releasing the workspace assigned by the builder). Some systems that implement the object-oriented paradigm provide finalizers to inform a system component that certain objects have completed their activity and can be collected and removed from the system (the mechanisms implemented by the Java-specific interpreter). A first distinction between relational type and object KRP systems is shown in Table 3.

Associations between classes may indicate: class interaction (an object of class A uses a class B object, for example a Subscriber borrows a Book, here classes are a Subscriber and a Book), multiplicities (a Book may have multiple editions or copies), roles, directions of message communication, aggregation and composition etc.

Aggregation is the method by which objects are allowed to incorporate other objects, i.e. the implementation of "xPy: the x relationship is part of y". Therefore, the fields of an object can be not only primitive data but also objects. In this way, objects with increasingly complex structures can be created.

Inheritance is a specific object-oriented mechanism that allows a class A to inherit the fields (data and methods) of a class B. It results, in fact, in a class hierarchy. We consider that if B inherits the properties of class A, then A is superclass for B, and B is a subclass of A. This chain can continue until the actual specialization. Switching from class to subclass is done by adding attributes and / or methods. For example, the Male class and the Female class both have all the attributes of the Human class, but each has specific attributes.

A technique specific to object processing is polymorphism, which allows the same operation to be performed by different algorithms in different classes. For example, if Circle, Triangle, and Square are subclasses of the Figure class, then the surface () method of the Figure class will be implemented differently in each sublayer.

Transforming an object-oriented database into an object-based knowledge base involves encapsulating multiple predicates to form an abstract object

Table 3: Relational and object oriented models. Comparison.

Relational Model	Object Oriented Model	Differences
Entity	Object	The object also includes behavior (by means of access, get, set) and processing. The entity refers only to the data fields (attributes, record, record, struct).
Attributes	Attributes	No
Types of Entities	Messages	The concept of message does not make sense in the relational model.
	Classes	The relational model does not work with classes.
Entities	Instances	The implementation of the relational model involves the use of structured type variables specific to a type definition, and in the object model, objects are called instances of the class. In situations where entities are objects, in the relational model, the relationship between objects is treated as any relationship. In the objectual model we can talk about the types of relationships: association, aggregation, composition, inheritance.
	Encapsulation	Concept valid only in object context.
	Builders (Constructors)/Destructor	Methods specific to the object-oriented model, but can be simulated in the dynamic implementation (of pointers) of entity-based processing techniques.

(a prototype) similar to a Prolog module. Thus, predicate definitions become methods. Some fields (attributes and methods) will be of the class (meaning they are usable by all members of the class). We exemplify object-oriented concepts by the following Prolog Objects specification:

Animal :: {}.

Bird :: super (animal) & wear (feathers) & lives (tree) & run (flying).

Penguin :: super (bird) & lives (earth) & run (goes) & run (swim) & size (medium).

By interrogation Penguin :: run (M), we get the results M = goes and M = swim.

The interrogation Penguin :: wear (S) gets S = feathers.

In the same way as the multiple inheritance in C ++ (or the Java version of classes and interfaces), multiple inheritance is extended in Prolog Sicstus (PO: Prolog Objects). Suppose we have definitions for sports classes and teacher. Then the statement for John as a simultaneous teacher and sportsman is specified by john :: super (sportsman) & super (teacher).

Therefore, object-oriented KRP systems allow the representation and processing of knowledge gained through association operations, the inheritance of all attributes or just a subset, and the hierarchical processing of knowledge through the polymorphism technique. The main stages of the object-oriented approach are: domain definition, object identification, behavioral identification, building objects, classifying objects (establishing sets of classes, subclasses, etc.), identifying collaborations between classes (various types of associations), defining restrictions, defining preconditions and postconditions to ensure logical correct modeling.

3. ***Concepts and associated graphs***

As we have described above, knowledge can be recorded at the time where the concepts were identified, and the relationships between them and the technical solution for computer implementation are known. In this section, we focus on graph based KRP systems. According to Jayaram et al. (2013), the interest in graphs associated with knowledge has increased greatly during last ten years. Examples include, but are not limited to, DBpedia [Auer et.al. (2007)], YAGO [Suchanek et.al. (2007)], Freebase [Bollacker et.al. (2008)] and Probase [Wu et.al.(2012)].

The main notions we will use are: node, link, special node (contextual behavior), contextual link, semantic (simple) network, conceptual graph, multilevel semantic networks, and process node.

A node is an elementary (indivisible) piece of knowledge (term, entity, concept) that is unique within the model. Therefore, nodes can be generic objects, things, events, actions, ideas, people, high-level concepts (such as abstraction) such as person, sport event, learning, feelings, activities, geometric figures, etc. More comprehensive knowledge units will be represented by groups of interconnected nodes. The smallest semantic difference between two terms will lead to the existence of two distinct nodes. Nodes have (monosemantic) names that label them. The nodes may have also other attributes. Thus, nodes can be of several categories: node (common node, static node, entity,

concept, term), contextual node (abstract, group, set, class, frame, type)) and process nodes (binding, kinship, dynamic, functional, action, conditioning).

The connection is designed to connect two nodes. There are only binary links. For example, for the sentence "University of Pitesti" two nodes will be created: one for "University" and the second for "Pitesti". Between these nodes will be a link (for example, with the arrow from the "University" node to the "Pitesti" node to eliminate the ambiguity of "Pitesti University".

A contextual link occurs between a contextual node and a specialized node, for example, from the node "Human" to the node "Male" or from the node "Human" to the node "Woman".

Conceptual charts were originally introduced to describe schemes used in databases, but then used in intelligent applications (artificial intelligence, cognitive sciences, etc.) [Sowa, 2008]. In the CSG (2000), a standard on graphical charts developed by the NCITS.T2 Committee for Interchange and Interpretation of Information is described. In the following, some remarks are given about the statement "Ionescu (as a Person) Goes to Pitesti (like City) by Bus". We note that the conceptual nodes "Goes" and "Bus" have no names, whereas the "Person" and "City" nodes have. Also, the links between the nodes have been marked by their roles: Agent, Destination, and Instrument.

Formally, in the logic language, we can say:
$(\exists y)(\exists y)$ (Goes $(x) \wedge$ Person (Ionescu) $\wedge$ City (Pitesti) $\wedge$ Bus $\wedge$ Agent (Ionescu, $x) \wedge$ Destination (x, Pitesti) $\wedge$ Instrument (x, y).

Conceptual graphs can be described in three ways: GDF (Graphical Display Form), CGIF (Conceptual Graph Interchange Format) or CLIF (Common Logic Interchange Format) and LF (linear form), any of which can be translated into KIF (Knowledge Interchange Format). We can assume that this approach of Sowa (2008) is based on formal logic, more precisely on predicate logic.

GDF representation uses rectangles to present concepts, circles or ellipses to introduce the relationship between concepts, as mentioned above. Arrows are inserted between relationships and concepts. If a relationship has more than two arcs then they will be numbered.

In the LF representation, the concepts are presented in square brackets [], and the relationships between round brackets (). Instead of the arcs in the graphical representation, in the LF representation a special symbol, for example Θ is used, as in the example: [Book] Θ (On) Θ [Table], which partially shapes the activity of the reading room of a library, where readers' places are relevant information in a given study. GDF and LF representations are not suitable for communication between computers, but only for human communication or for communication between man and computer.

The CGIF representation uses concept identification tags used in the description of the relationship using the binding process (see CGS (2000)): [Book: $*X$] [Table: $*Y$] (On $?X$ $?Y$) or simpler [Book] [Table] without specifying generic instances. It is obvious that the LF and CGIF representations are logically equivalent, being different only as a representation style.

For the situation where the internal representation of knowledge is not the conceptual graph, but primary data structures such as C ++ structure / class, the extended KIF formalism can be used: (there is (?X Book) (?Y Table) (On ?X ?Y)).

We note that all of the above descriptions can be interpreted using the predicate calculation: ($\exists x$) Book (x) ($\exists y$) Table (y) On (x, y). We note that in the GDF representation, the Goes concept is related to three other concepts: Person, City and Bus. In this case, a modified LF representation (in multiline format) is:

Goes :-
    Agent (person: Ionescu)
    Destination (City: Pitesti)
    Tools (Bus),

what in CGIF is represented by:

Goes: $*x$
    [Person: Ionescu * y]
    [City: Pitesti * z]
    [Bus: a]
    (Agent ?X ?Y) (Destination ?X ?Z) (Instrument ?X ?Z),

which is equivalent, in KIF representation, to: (There is (?X Goes) (?Y Person) (?Z City) (?A Bus)) (and (Name ?Y Ionescu) (Name ?Z Pitesti) (Agent ?X ?Y) (Destinatie ?X ?Z) (Instrument ?X ?A)), equivalent also, in predicate calculus, with the description given above.

4. ***Semantic networks***

The concept of a semantic network or associative network was introduced approximately between 1965-1970. The purpose of the first semantic network was to represent sentences from natural language. Being a graph-based representation, a semantic network takes the form of an ensemble of knots and arcs, oriented and labeled. Nodes represent objects subject to representation and interpretation. An object may be an abstract or specialized concept, may be an attribute, etc. Arcs are used to represent the links that exist between objects.

A semantic network (SN) is an assembly (G, L, T, ϕ, f, S, g) where:

- $G = (V, L_0, T_0, f_0)$ is a labeled oriented graph such as:
  - V is the set of node labels,
  - L_0 is a distinct set from V and is the set of arc tags,
  - T_0 is a non-empty set of binary relations on V, and
  - f_0 is a surjective function from L_0 to T_0: The nodes labeled with x and y are connected by the arc labeled with a if and only if $(x, y) \in f_0(a)$;
- L is a set containing L_0;

- T is a set containing T_0, but included in the transitive closure of T_0, i.e. it comprises those relations that are formed, starting from relationships in T_0, only by the relationship-composition operation;
- ϕ: $LxL \to L$ is a partially defined map;
- $f : L \to T$ is a surjective extension of f_0 such that $f(\phi(a, b)) = f(a) \circ f(b)$;
- S is a set that represents the semantic space;
- $g : VxLxV \to S$ is called semantic map and $g(x, e, y)$ represents "semantics" associated with the fact that the nodes x and y are connected by the arc labeled with e.

For two nodes labeled with x and y, the answer to a query related to x and y can be formulated only if there is a path from x to y, denoted by $x = x_1, x_2, ..., x_n = y$, with the arcs labeled by $e_1, e_2, ..., e_{n-1}$, the associated semantics being $g(x, \phi(...\phi(e_1, e_2), e_{n-1}), y)$.

Implementing a query on two nodes in PROLOG calls for the predicate associated with g. More precisely, semantics (X, Y, A) is a predicate that solves $g(X, A, Y)$.

In the following, we will illustrate a way of processing queries in conceptual graphs, easily adaptable to semantic networks, for IT implementation.

We will consider multigraphs, that is, directed graphs, which allow for the existence of several links between two nodes, each with its label. We will differentiate between the query and response example, these being described by tuples, a tuple being an ensemble of entities. Given a multigraph G and a tuple query t, we want to identify k-tuples with the highest similarity to t (obviously in decreasing order). By solving this problem, we can communicate with a multigraph based KRP system, giving it an example of what we want, and the KRP system provides us with the answers found after visiting the knowledge base stored in the given graph.

To solve the task, it is useful to consider the neighborhood graph of the threshold d associated with a tuple t, denoted by H_t, with the nodes $V(H_t)$ and the edges $E(H_t)$. This graph is associated with the user's intention. The graph associated with a query is a weakly connected subgraph, where there is a chain between any two nodes (no matter the meaning of the link) associated with H_t and it contains all the query instances, which are denoted by Q_t. A response graph A associated with a query graph Q is an isomorphic graph with Q. Obviously, for a graph Q there may be more response graphs belonging to the set $A(Q)$. It is useful to find *the maximum query graph* having m edges. A method for solving this problem is described by the following procedure (see Jayaram et al. (2013)):

Procedure MQHT
Input Parameters: The neighborhood graph H_t, the query tuple t, an integer d
Output Object: MQH_t graph
The steps of the algorithm:

1. $M = d/(|t|+1)$; $V(MQH_t) = \emptyset, E(MQHt) = \emptyset; G = \emptyset$;
2. For each v_i of t execute
   a) Gv_i = the subgraph obtained through the depth search
      containing all the vertices connected throw v_i in t;
   b) Add Gv_i to G.
3. G_0 = the graph obtained through depth search
      associated with querry entities. Add G_0 to G.
4. For every graph g of G execute
   4.1. $p = 1; s_1 = 1; s = m$;
   4.2. Why $s > 0$ do
      4.2.1 M_s = the weakly connected component of the graph G
         containing the edges of G of level s,
         relative to the given queries;
      4.2.2 If M_s is non-empty then
         4.2.2.1 If M_s has exactly m links then proceed to step 4.3.
         4.2.2.2 If M_s has less than m connections then
            [$s_1 = s$; If $p = -1$ then proceed with step 4.3]
         4.2.2.3 If M_s has more than m connections then
            a) if $s_1 > 0$ then $s = s_1$; Continue with step 4.3
            b) $s_2 = s$; $p = -1$.
      4.2.3 $s = s + p$;
   4.3 If $s = 0$ then $s = s_2$;
      Add $V(M_s)$ to $V(MQH_t)$; Add $E(M_s)$ to $E(MQH_t)$.

Remark 1) Semantic networks highlight lexical, structural, procedural and semantic aspects. The lexical component addresses the allowed symbols in the formation of node names, labels, etc. The structural component refers to the possible interconnection restrictions of the nodes. The procedural component addresses building operations, read access, write access, and removal procedures. Obviously, the semantic component allows deduction of the interpretation of the path between two nodes of the network.

Remark 2) In order to process information stored in concept graphs, it is necessary to apply a depth first algortithm. Hence, the use of greedy strategy in determining weakly related subgraphs is a first choice from an algorithmic point of view. Optimizing the algorithm implementation is an important requirement.

5. ***Conclusions***

This paper had presented some traditional approaches in knowledge representation. Based on graphical diagrams or mathematical models like relations, graphs, and semantic networks, the designer of a KRP system will consider formal representations in order to be processed by various parsers of file formats. Not only object oriented representations but also concept graphs can be used to describe knowledge to be processed by a future expert system.

References

1. Auer S., Bizer C., Kobilarov G., Lehmann J., R. Cyganiak R., Ives Z., *DBpedia: A Nucleus for a Web of Open data*, "ISWC", 2007.
2. Bollacker K., Evans C., Paritosh P., Sturge T., Taylor J., *Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge*, "SIGMOD", 12471250, 2008.
3. Borgida A., *Knowledge Representation Meets Databases A View of the Symbiosis*, "Proceedings of the 20th International Workshop on Description Logics DL'07", Bolzano University Press, http://dl.kr.org/dl2007/dl-workshops/dl2007/invited_3.pdf, 2007.
4. Ensing M., Paton R., Speel P.-H., and Rada R., *An Object-oriented Approach to Knowledge Representation in a Biomedical Domain*, "Artificial Intelligence in Medicine", 6, 459-482, 1994.
5. Jayaram N., Khan A., Li Ch., Yan X., Elmasri R., *Querying Knowledge Graphs by Example Entity Tuples*, arXiv: 1311.2100v1 [cs.DB], 2013.
6. Motro A., Yuan Q., *Querying Database Knowledge*, "Proceedings of ACM-SIGMOD International Conference on Management of Data", 173-183, 1990.
7. Novak J.D., Canas A.J., *The Theory Underlying Concept Maps and How to Construct and Use Them*, Technical Report IHMC CmapTools 2006-01 (Rev 2008–01), http://cmap.ihmc.us/docs/theory-of-concept-maps, 2008.
8. Sowa, J.F., *Semantic Networks*, "Encyclopedia of Artificial Intelligence" (ed. S.C. Shapiro), Wiley and Sons, 1987.
9. Sowa J.F., *Conceptual Graphs*, "Handbook of Knowledge Representation" (eds. F. van Harmelen, V. Lifschitz, and B. Porter), Elsevier, 213237, 2008.
10. Suchanek F.M., Kasneci G., Weikum G. 2008. *YAGO: A Large Ontology from Wikipedia and Wordnet*, "Web Semantics: Science, Services and Agents on the World Wide Web", 6(3), 203217, 2008.
11. Wu W., Li H., Wang H., Zhu K.Q., *Probase: A Probabilistic Taxonomy for Text Understanding*, "SIGMOD", 481492, 2012.
12. ***, *Conceptual Graph Standard*, NCITS.T2 Committee on Information Interchange and Interpretation, http://www.w3.org/DesignIssues/Sowa/cgstand.htm.
13. ***, *Prolog Objects (PO)*, https://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_35.html.
14. ***, *CMAP*, http://cmap.ihmc.us/cmaptools/cmaptools-download/.