

Semantic Subtext in the Bubble Sort Algorithm

Olegs Verhodubs

oleg.verhodub@inbox.lv

Abstract. The bubble sort algorithm is one of the simplest and most well-known in programming. That is why this algorithm was chosen as an experimental algorithm for studying the meanings embedded in it and their transformation. This research is part of a broader study whose final goal is to develop a system for the automatic generation of algorithms. Automatic algorithm generation can be implemented in various ways, but in this case, preference is given to a semantic approach — more precisely, an approach based on Semantic Web Technologies. The advantage of this approach lies in the direct, “willful” movement (from one concept to another) towards the result, as well as in the ability to trace the entire path from the initial data to the generated algorithm. To achieve the final goal, it is proposed to develop an ontology of the semantic subtext of the algorithm. This ontology will serve as a source of rules for an expert system for the automatic generation of algorithms.

Keywords: Algorithms, Bubble Sort Algorithm, Algorithm Generation, Python, Ontology, Rules from Ontology, Semantic Web

I. Introduction

Philosophical thought has been developing for many centuries, but the ancient Greek philosopher Plato was and still remains one of the greatest philosophers of all time. Many hundreds of years have passed since his life and death, and over this entire time we have grown accustomed to Plato’s idea that, in addition to the material world, there is also an immaterial world — namely, the world of ideas. According to Plato, the world of ideas is the world of ideal prototypes, that is, the models of all things, while the world we observe is merely a pitiful copy of some absolutely true being that is inaccessible to our senses [1]. The Platonic world of ideas has never been detected in its pure form, but a certain scientific interpretation of the concept of the immaterial world, close to Plato’s concept, is possible even at the current level of civilization’s development. For example, ideas or meanings physically exist, as they can be detected in the brain of every person [2]. These ideas are subjective in nature, as they differ to some extent from person to person. However, on the whole, this subjectivity disappears and objectivity emerges as the number of subjects who have this particular idea or meaning increases. It can be said that objectivity is a function of subjectivity, and as the number of subjects grows, objectivity increases. Thus, we can assert that the immaterial world or the world of ideas exists and represents the sum of all the representations of the subjects. Moreover, this non-material world is objective, as it correlates with objective reality (at least at the level at which we understand this objective reality).

If we are talking about science, we are talking about knowledge. But if we are talking about technology, we are talking about algorithms. In this regard, algorithms are very important and are the cornerstone of our entire civilization, if we consider it from a technological perspective. Algorithms, of course, have always existed, but they became firmly embedded in our lives with the advent of computers, and now, when we talk about algorithms, we primarily imagine some code, or a computer program in one of the programming languages. Code is an algorithm in one of the programming languages, and outwardly it is a set of some actions. However, “under the hood,” that is, inside, each algorithm works with meanings, their constraints, and transformations. To demonstrate all of this clearly, one of the most well-known algorithms in

programming was chosen — namely, the bubble sort algorithm written in Python. This algorithm was analysed and thoroughly described from the perspective of the meanings embedded in it. This research is not purely academic, as it also potentially has practical interest — as a step towards the possibility of automatically generating algorithms. Moreover, unlike the “unconscious” automatic generation of algorithms through the use of artificial neural networks, in this case we are talking about the “conscious and volitional” generation of an algorithm, where the result — that is, the finished algorithm — emerges as a result of processing and transforming meanings. And although fully automatic generation of an algorithm of any complexity using this method is still a matter of the distant future, this research, in a sense, clarifies this path, which differs from other methods — including the method based on artificial neural networks.

This paper is organized into several sections. The next section semantically analyzes the bubble sort algorithm. The third section is intended to describe the ontology that needs to be created to enable the automatic generation of an algorithmic solution for a given task. The final section presents the conclusions, summarizing the study and outlining future practical applications—specifically, the development of an algorithm based on problem conditions and an ontology, without human intervention.

II. Semantic analysis of the bubble sort algorithm

Any object of observation can be considered from different perspectives. An algorithm is no exception; it can be viewed in various ways. For example, an algorithm, like any other system, can be studied either knowing its internal structure or without knowing it. These are models of systems that are well-known from cybernetics and the systems approach — namely, the black-box and white-box models. In turn, knowing the internal structure, or, better said, the internal composition of an algorithm, one can also interpret it in different ways. For example, one can analyze an algorithm by examining its composition line by line. Or there is the possibility to transform an algorithm written using a functional approach into an algorithm that produces identical results, as if it had been developed using an object-oriented programming approach. Another possibility is to consider the algorithm from the perspective of its essence, that is, from the perspective of the meanings embedded in this algorithm and their movement. This method is more preferable than the others, as it is natural and, so to speak, starts “from the ground up”. “From the ground up” here means that the algorithm is described starting from the concepts that are natural for it and their limitations. In other words, the algorithm is considered from the perspective of semantics. To demonstrate the semantic foundation, or semantic subtext, of an algorithm, let us take one of the most well-known algorithms — namely, the bubble sort algorithm implemented in Python. The semantic subtext of an algorithm is the sum of all the meanings embedded in that algorithm. The familiarity and simplicity of the bubble sort algorithm, as well as the simplicity and widespread use of the Python programming language, will make it easier to understand its semantic subtext. In general, programming is akin to art, so the same result can be achieved in different ways. Moreover, the same algorithm in its essence can be implemented with some differences. One implementation of the bubble sort algorithm for ascending order might look like this:

```

1 numbers = [5, 1, 4, 2, 8]
2
3 n = len(numbers)
4 for i in range(n):
5     swapped = False
6     for j in range(0, n - i - 1):
7         if numbers[j] > numbers[j + 1]:
8             numbers[j], numbers[j + 1] = numbers[j + 1], numbers[j]
9             swapped = True
10    if not swapped:
11        break
12
13 print(numbers)

```

Fig. 1. Text of the bubble sort program.

Figure 1 shows the bubble sort algorithm in ascending order, which is written in Python, but the essence of this algorithm can be briefly described in ordinary words without going into the implementation:

for all values in the set of values, swap two adjacent values if the current value being checked is greater than the next value, until no adjacent values are swapped during the previous pass through all the values in the set.

From this description of the essence of the bubble sort algorithm in ascending order, we can attempt to generate the program text in the Python programming language, as if we had not seen the presented program text in Fig. 1. This technique will allow us to structure the semantic subtext of the bubble sort algorithm.

So, from the first phrase “for all values in the set of values,” it follows that several values will be stored. Python has several data structures designed to store multiple values, namely: list, tuple, set, and dictionary [3]. Next comes the phrase “to swap two adjacent values,” which means that the order of elements in the data structure is important, and also that elements can be replaced. It follows that the tuple data structure is excluded, since the tuple data structure does not allow changing the values of elements. As for other data structures, for example, a dictionary, each element of the dictionary contains a pair: a key and a value. In our task, there is no need to store the key as well; this would be redundant, so this data structure is excluded. The set is also not suitable for our task, since the property of ordering is not inherent in it at all, that is, the order of the elements in the set is not defined. Thus, only the list remains.

Next, the phrase from the description of the algorithm is “for all values from the set of values, swap...” means that you need to process each value from the entire set of values, which means that you need to use a loop. Of course, the cycle cannot work if it is not known how many times certain actions need to be repeated. The phrase “for all values from a set of values, swap” makes it clear that the operation needs to be performed as many times as there are values in the set of values.

Another important point. The phrase “until no adjacent values are swapped during the previous pass through all the values in the set” implies that some information about the processing of all values must be stored in some way — namely, information about whether an exchange of adjacent values occurred during the current pass. Any information in the algorithm is stored

using variables, so an additional variable is needed to store the fact that an exchange of values occurred in the set of values. Moreover, it follows from the above phrase that there may be several passes of all values, which means that there is a need for another cycle. In addition, one cycle must depend on another cycle, which means that one cycle must be nested in another.

So, we can conclude that the semantic implication of the bubble sorting algorithm is based on the properties of its components, namely variables, cycles, conditions, etc. It is the properties of the components of the algorithm that are the manifestation of internal constraints underlying the underlying meanings. Thus, while working with the properties of the algorithm's components, we are indirectly working with the meanings contained in the algorithm itself.

III. Semantic subtext algorithm ontology

In terms of its composition, the bubble sort algorithm is no different from any other algorithm, as it consists of the same variables, loops, conditions, and other elements. Due to the uniformity of the composition of all algorithms, it is possible to compile an ordered set of all possible algorithm elements with a focus on semantics. A detailed description of the semantic subtext of each element from the ordered set of all possible algorithm elements will, on the one hand, provide a clear understanding of why a particular algorithm element is chosen at a given point in the algorithm, and, on the other hand, create the prerequisites for implementing a tool for developing algorithms without human involvement.

Ontologies are a universal means of representing the semantics of any subject area [4], and therefore they are the most suitable way to describe all possible elements of an algorithm. To create such an ontology, preference should be given to the OWL (Web Ontology Language) [5], which is the de facto standard for describing any subject area in the form of an ontology. When describing algorithm elements, attention should be paid to the key differences between algorithm elements, especially between similar, or “related” algorithm elements. The key differences between the elements of an algorithm will provide us with criteria that can be used to guide the selection of a particular element for building an algorithm in automatic mode. For example, a list and a tuple in the Python programming language can be called “related,” since both a list and a tuple are designed to store not one, but several values. If we ignore the syntactic differences between a list and a tuple in Python, the key difference between them is the possibility of modifying values in a list, which is not possible in a tuple. Thus, visually, a rough fragment of the ontology that shows the differences between the list and tuple types in the Python language can be represented as follows (Fig. 2):

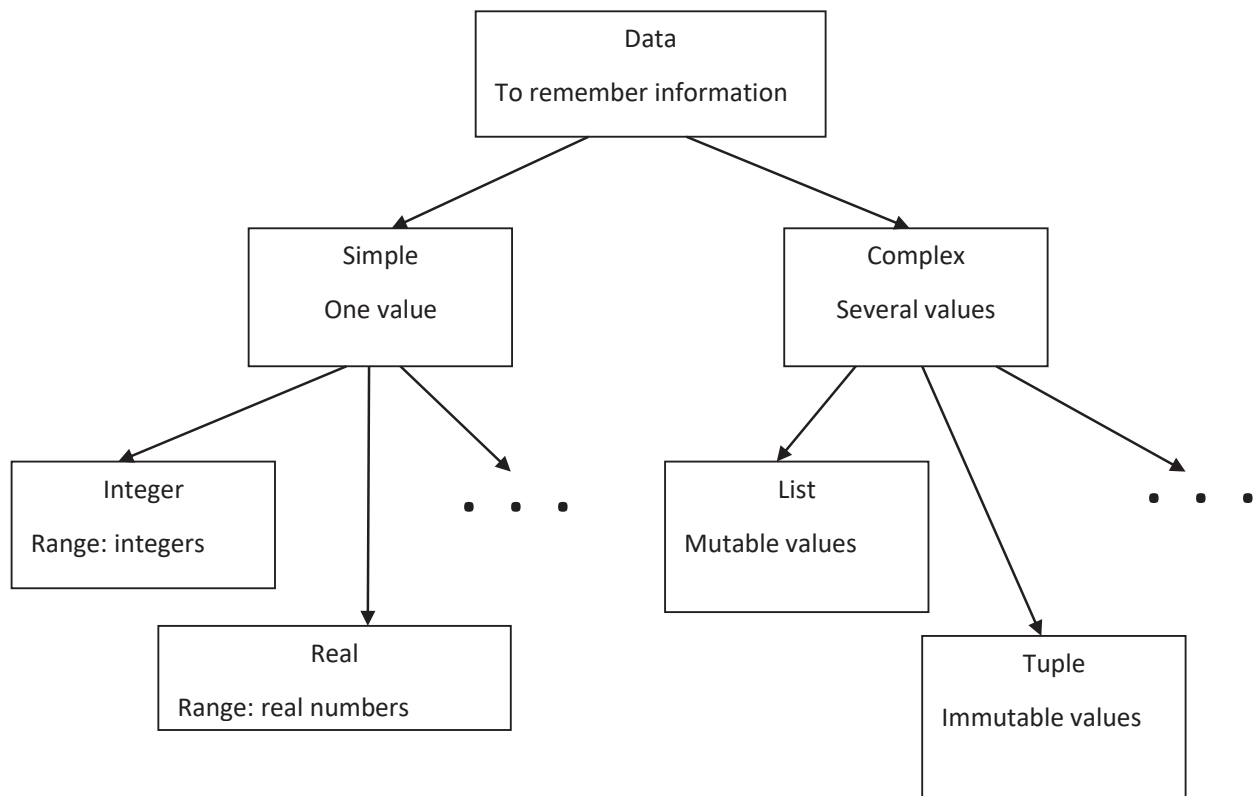


Fig. 2. A fragment of the ontology showing the differences in the types of list and tuple.

Figure 2 shows just a small fragment of the ontology that shows the differences in the list and tuple types, as well as the differences in the integer and non-integer types in Python. But, of course, this ontology should include all possible elements of the algorithm, not just types, in order to clearly identify the key differences between all possible elements of the algorithm.

Any ontology, including the ontology of the semantic subtext of an algorithm, can serve as a source for generating rules from it [6], [7]. Just as a certain feature or several features characterize an object in the real world, so one property or several properties of a class in an ontology characterize the class itself [6]. If each element of the algorithm is represented as a class, then we get a rule that determines the choice of one or another element of the algorithm. For example, from the presented fragment of the ontology in Fig.2. It is possible to generate, among other things, such rules related to the choice of a particular type of data for storing some information:

If you need to store a single value, then choose from simple types.

If you need to store multiple values, then choose from complex types.

If you need to store mutable values, then choose a list.

If you need to store immutable values, then choose a tuple.

and so on...

Having rules generated from the ontology, it is possible to build an expert system based on them, which will be capable of automatically and iteratively selecting one or another element of the automatically generated algorithm. That is, we are talking about the automatic construction of an

algorithm using an expert system based on rules generated from the ontology of the semantic subtext of the algorithm. Another issue should be addressed. From the perspective of the future implementation of an automatic algorithm construction system, one might draw the erroneous conclusion that the rules for selecting a particular algorithmic element at a particular stage of work are what matter, and the ontology of the semantic subtext of the algorithm appears to be an unnecessary component. However, this is not entirely accurate, because any ontology in general, and the ontology of the semantic subtext of an algorithm in particular, is not just a space, but also a structure of meanings, which makes it possible to conduct an in-depth analysis of the subject area of algorithms with the prospect of revealing non-obvious dependencies and interrelationships among the elements of the algorithm. Therefore, the ontology of the semantic subtext of an algorithm is not an unnecessary component; on the contrary, it serves as a technologically convenient means for further processing of the meanings embedded in the algorithm and their structured representation.

IV. Conclusion

In this paper, the bubble sort algorithm in Python was analyzed in detail and described from a semantic perspective. This description showed that the choice of each programming language construct in the algorithm is determined by clear criteria. These criteria for choosing a particular language construct at a given point stem from the unique properties that characterize each of all possible constructs in the Python programming language. The properties of all possible constructs of a specific programming language should be described in the form of an ontology in the OWL language. In this paper, this ontology is called the ontology of the semantic subtext of the algorithm. Specifying all possible constructs of a programming language using OWL allows for an in-depth analysis of this programming language, identifying similarities and differences with other languages, pinpointing bottlenecks in the programming language, etc. But even more importantly, it is possible to generate rules from any ontology and then use these generated rules in an expert system for developing the required algorithm. And although expert systems often work in conjunction with a human, an expert system for developing the required algorithm can be fully automatic, receiving only the initial data from the user, as well as a description of the required result. There are still several fundamental issues that need to be addressed for the successful implementation of such an expert system; however, it appears that there are no insurmountable obstacles along the way.

References

- [1] Available online: https://philosophy_for_students.academic.ru/356/Мир_идей_%28Эйдосов%29_Платона
[Accessed: October 05, 2026]
- [2] N. Doidge, The Brain That Changes Itself: Stories of Personal Triumph from the Frontiers of Brain Science. New York: Viking Press, 2007 (in Russian).
- [3] D. Bader, Python Tricks: The Book, 2018 (in Russian)
- [4] J. Davis, R. Studer, P. Warren, Semantic Web Technologies Trends and Research in Ontology-based Systems. Chichester: John Wiley & Sons Ltd, 2006.
- [5] OWL Web Ontology Language Reference. Available online: <https://www.w3.org/TR/owl-ref/> [Accessed: October 06, 2026]
- [6] O. Verhodubs, J. Grundspenkis, Evolution of ontology potential for generation of rules, 2012.
- [7] O. Verhodubs, Ontology as a source for rule generation, 2014.