Konuralp Journal of Mathematics, 6 (1) (2018) 1-6

Konuralp Journal of Mathematics

Journal Homepage: www.dergipark.gov.tr/konuralpjournalmath
e-ISSN: 2147-625X

Inference situations, Counter-Model Constructions and A Computer Implementation of A Logic Composed of Intersecting Adjectives and The Quantifier "More"

Selçuk Topal[1*] **and Yasin Akünsoy**[2]

[1]*Department of Mathematics, Faculty of Arts and Sciences, Bitlis Eren University, Bitlis, Turkey*
[2]*Department of Mathematics, Faculty of Arts and Sciences, Bitlis Eren University, Bitlis, Turkey*
[*]*Corresponding author E-mail: s.topal@beu.edu.tr*

Abstract

In this paper, we study the logic of language of $L(More, IA)$. The logic contains the quantifier called '' more '' which makes cardinality comparisons can not be expressed in the language of the first order logic. The sentence forms are basically the form of ''There are more y than x .'' with x and y being common plural nouns. The sentence forms of common plural nouns combined with intersecting adjectives are ''There are more b y than a x." with the intersecting adjectives a and b. We focus on derivation algorithms of the sentences having this type of quantifier and algorithms of construction of counter-models when the derivations are not provided.

Keywords: *Cardinality comparison, graph algorithms, intersecting adjectives, logic of natural languages*
2010 Mathematics Subject Classification: *05C85, 03B65*

1. Introduction

The fundamental form of human reasoning and intelligent systems is syllogism. First examples of syllogisms, which were initially called logic in language, were given by Aristotle [1]. The Aristotelian syllogism consists of simple sentences with the quantifiers all, some, and no [1]. In the modern sense, set sizes (element count comparison) or cardinality comparisons of sets are included. It is set interpretations of plural names within sentences that are intended to be compared from the sets. Sentences including "more" can not be expressed on the first order logic. For this reason, working with these types of logic involves some difficulties and has a significant place in the field of logic. Intersecting Adjectives in syllogistic logic have been formally used by Moss for the first time [2]. Later, Moss introduced some of the logics that Aristotelian syllogisms and cardinality comparisons together evaluated on finite models [3]. Moss and Topal introduced a logic of syllogistics (countable or uncountable) models, comparisons of the cardinality, and sets of of interpretations of nouns on infinite sets[4]. The logic in this paper consists of sentences with "more" and intersecting adjectives, is the first time described in here.

Remark 1.1. *This paper is extracted from master thesis entitled "Algorithmic Analysis and A Computer Implementation of A Syllogistic Logic Composed of Cardinality Comparisons of Nouns and Intersecting Adjectives", (Yasin Akünsoy, Master Thesis, Bitlis Eren University, Turkey, 2017).*

2. The Logic $L(More, IA)$

Let P be a set of plural nouns $x, y, z, ...$ (as variables). Sentences in the language form $\exists^{>}(x,y)$, and read as " There are more x than y" . Semantics is constructed on a finite set Γ of sentences and finite universe M. An interpretation function $[[]]$ is defined from P to M and for all $x \in P$, $[[x]] \subseteq M$. Cardinality of $[[x]]$ is denoted by $|[[x]]|$. This language is called $L(More)$. Figure 1 shows the rules of $L(More)$. We introduce $L(More, IA)$ by adding intersective adjectives to $L(More)$. Colors can be good examples for intersecting adjectives (IA) such as blue, green, black, ... and plural nouns are such as cats, cars, machines, ... Syntax in this language contains nouns as $x, y, z, ...$ and intersecting adjectives as $a_1, a_2, a_3 \ldots$. A complex noun ax is a noun where x is a basic noun and a is an intersecting adjective. Variables x, y, z will be basic nouns ve p, q, r will be nous (basic or complex). Semantics is on finite sets M and, $[[x]] \subseteq M$ for all basic noun x. The interpretation of ax is $[[ax]] = [[a]] \cap [[x]]$. For a model $\mathscr{M} = (M, [[]])$ " $M \models \exists^{>}(p,q)$ is true if and only if $|[[p]]| > |[[q]]|$ in $\mathscr{M}$.

$$\frac{\exists^{>}(n,p)\ \exists^{>}(p,q)}{\exists^{>}(n,q)}\ (\mathrm{tr}) \qquad \frac{\exists^{>}(p,p)}{\varphi}\ (\chi 1)$$

Figure 2.1: Rules of $L(More)$.

$$\frac{\exists^{>}(n,p)\ \exists^{>}(p,q)}{\exists^{>}(n,q)}\ (\mathrm{tr}) \qquad \frac{\exists^{>}(ax,q)}{\exists^{>}(x,q)}\ (\mathrm{ia1}) \qquad \frac{\exists^{>}(p,x)}{\exists^{>}(p,ax)}\ (\mathrm{ia2})$$

$$\frac{\exists^{>}(p,p)}{\varphi}\ (\chi 1) \qquad \frac{\exists^{>}(redx,x)}{\varphi}\ (\chi 2)$$

Figure 2.2: Rules of $L(More, IA)$.

The rules (χ_1) and (χ_2) are called contradictions in Figure 2.2. The rules say that if *more p than p* or *more ax than x* can be derivable from a proof tree, then the model is inconsistent and therefore, one can derive all possible sentences from this model. The rules (ia1) and (ia2) in Figure 2.2 contains intersecting adjectives. The rule $(ia1)$ says that if *more a x than q* can be derived from a proof tree, then *more x than q* is derivable. The rule (ia2) says that if *more p than x* is derivable in a proof tree, then *more p than ax* can do. Finally, the rule (tr) is called transitive and says that *more n than p* and *more p than q* can be derived, then we have *more n than q* .

3. Inferences and Counter-Models

In this section, we consider inferences and counter-models in $L(More, IA)$.

3.1. Inferences

The rule (tr) is used in inference (1). (tr) is a transitive relation.

There are more students than teachers.
There are more persons than students.
—————————————————— (1)
therefore, There are more persons than teachers

A directed graph representation for inference (1) is given in figure 3.1. (a) shows the premises and (b) shows is a direct application of the transitivity.
There are more green bicycles than red cars.
There are more blue tractors than green bicycles.
—————————————————— (2)
Therefore, there are more tractors than red cars.

In inference (2), firstly by (tr) *there are more blue tractors than red cars* and secondly by (ia2), we have *there are more tractors than red cars*. For easier understanding of the inferences and transferring to computer as data, L (More, IA) can be represented by the labeled graph theoretical structures. [5]. In addition, there is no need for a labeling procedure since only sentences with "more" are worked on.

Figure 3.2 gives the theoretical representation of the application for inference (2). (a) for (tr), and then the application of (b) for (ia2) are represented. Although the order of application of the rules is sometimes not important, we note that it is important here.

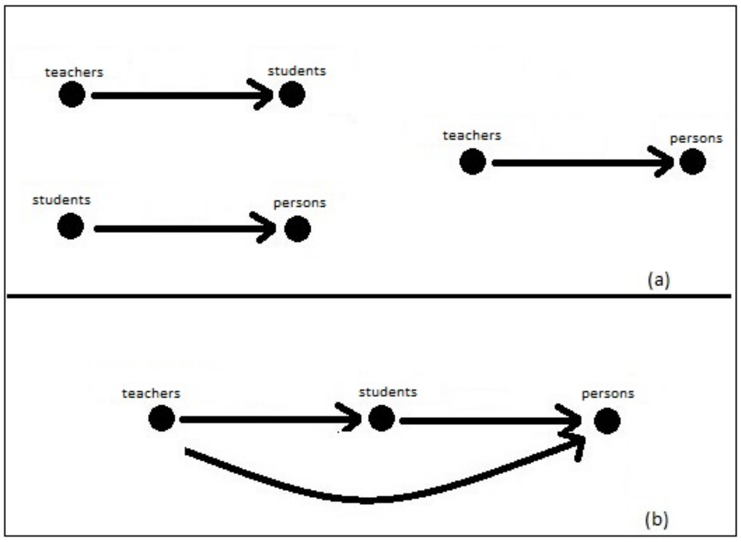


Figure 3.1: A directed graph representation for inference (1).

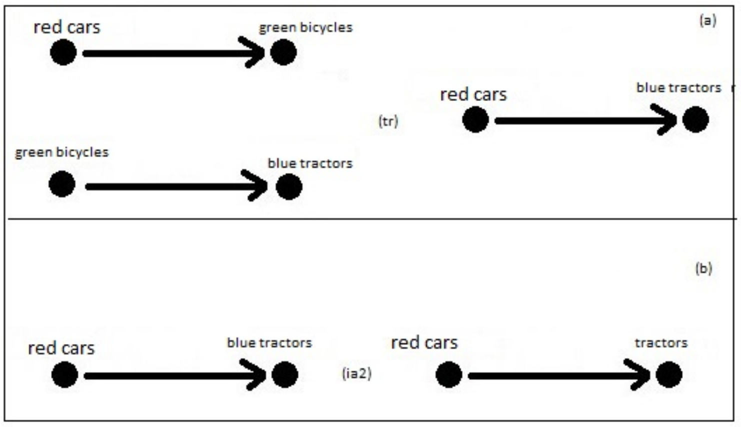


Figure 3.2: A directed graph representation for inference (2).

3.1.1. Inference algorithms

In this section, we explain how to make the question of whether a sentence can be derived from a set of sentences given in the language of the logic. Our main goal is to get all the sentences that can be derived from a given set of sentences. The first issue here is to determine if the set of sentences of the algorithm is consistent.
As seen in Figure 2.2, there are two types of inconsistency, (χ_1) and (χ_2). (χ_1) says that no cardinality of a set interpretation of a nouns can be more that itself. First of all, remember that we assign a directed graph for each of the sentences of the logic, and then we define the loop.

Definition 3.1. *Let G=(V,E) be a directed-graph. A loop is an edge e of G whose endvertices are the same vertex [6].*

If a set Γ of sentences given in Definition 3.1 contains "more v than v" or "more v than v" can be derived from Γ, - will be inconsistent.
If there is a path $v \longrightarrow v$ with the representation of the graph, then a loop is found and the inconsistency is reached. Of course, this loop applies not only to sentences containing nouns of type x, y and z but also to sentences containing intersecting adjectives of the form *ax*, *by* and *cz*.
It is necessary to make a forward search to get these types of sentences. For this, we need to obtain the inferences by applying the rules (tr), (ia1) and (ia2) to Γ.

Example 3.1. *For a given* $\Gamma = \{$ *more v than w, more than h, more h than v* $\}$ *is found to be inconsistent because we can obtain* $v \longrightarrow w \longrightarrow h \longrightarrow v$, *ie* $v \longrightarrow v$.

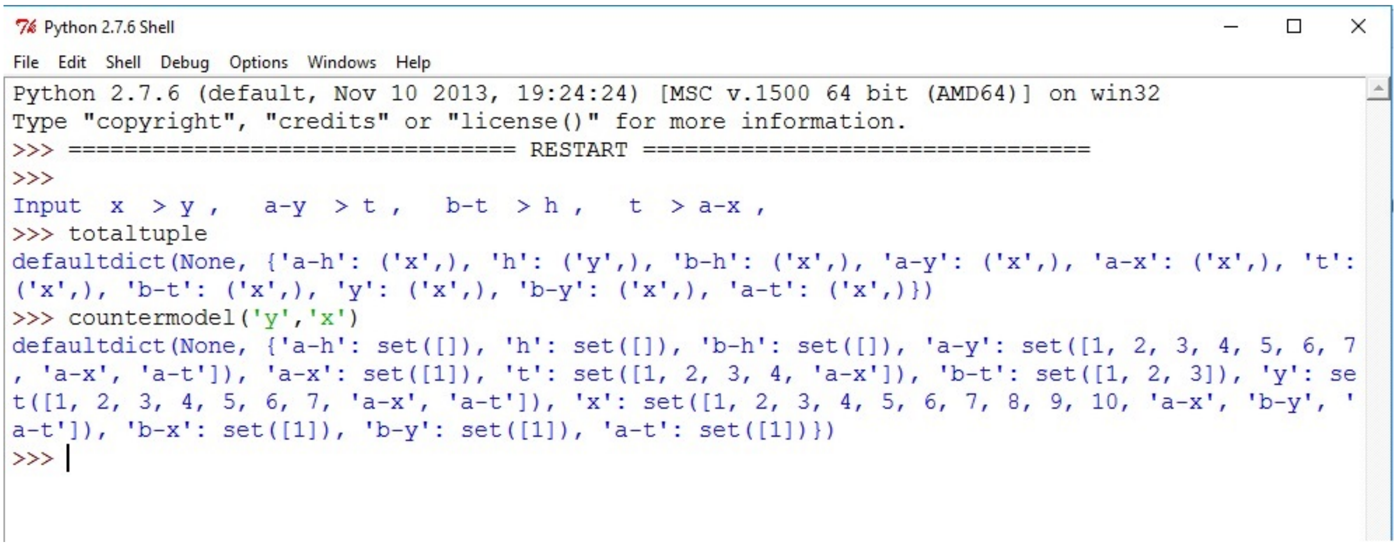
For contradictions finding via the rule (χ_2), a given Γ must contain "more a x than x" or be derived from Γ. Controlling whether or not there is a path of $ax \longrightarrow x$ is sufficient to achieve the inconsistency after having obtained the rules (tr), (ia1) and (ia2) derived from Γ.

Example 3.2. *For a given* $\Gamma = \{$ *more a x than w,more w than h, more h than x* $\}$ *is seen to be inconsistent because we can obtain* $ax \longrightarrow w \longrightarrow h \longrightarrow x$ *and* $x \longrightarrow x$.

Basically, all the derivations obtained from a set of given sentences by transforming the language of the rules in Figure 2.2 into a directed graph structure. Each path obtained during this application, that is, every noun (each vertex) that reaches each other is added to the graph, so the set of derivations to be created is brought to the graph. Therefore, it would be sufficient to check whether a given sentence belongs to a set of all derivations in order to determine whether or not a given sentence can be derived. If the set is inconsistent, the process can be terminated without any additional processing.

3.2. Counter-models

In this section, we will focus on the counter-models. Let's first explain what the counter-model is. A set Γ of sentences and a sentence of β can not be derived from Γ are given. In this case, we will construct a model that makes correct all the sentences in Γ, but makes β false. Let's make the concept clearer with a simple example.
For given set of sentences, $\Gamma = \{$ *more v than w, more than h, more h than k* $\}$, can be easily seen that the sentence of "more k than v" can not derived from the Γ. Now let's create a model that makes correct sentences of Γ but falsifies sentence of "more k than v". The model's universe is $M = \{0,1,2,3\}$ and $[[k]] = \{3\}, [[h]] = \{0,2\}, [[[v]] = \{0,1,2,3\}$. Note that $[[k]]$ has one element, $[[h]]$ has two elements, $[[w]]$ has three elements and [[v]] has four elements.
When we consider these nouns in terms of their cardinalities, it is seen that all Γ sentences are true in this model. On the other hand, the sentence of "more k than v" is false in this model because $[[k]]$ has a single element while $[[v]]$ has four element. So we built an counter-model for a sentence that is not derived from the set. The set of sentences given in Example 3 is simple and construction of the model is easy. If there were more sentences and there were intersecting adjectives in the nouns of these sentences, the construction of the counter- model would be much more complex and confusing. In particular, the intersection of the sets in the sentences containing the intersecting adjectives is a matter of concern, so more attention should be paid to the construction of the model. If $[[ax]]$ element contains an element, then naturally $[[x]]$must also contain this element. Otherwise, a sentence belonging to Γ may be false. For the sake of clarity, we will describe the counter-model algorithm with a sample output of the program on the Python [7] platform. It can be reached to program address of [8].

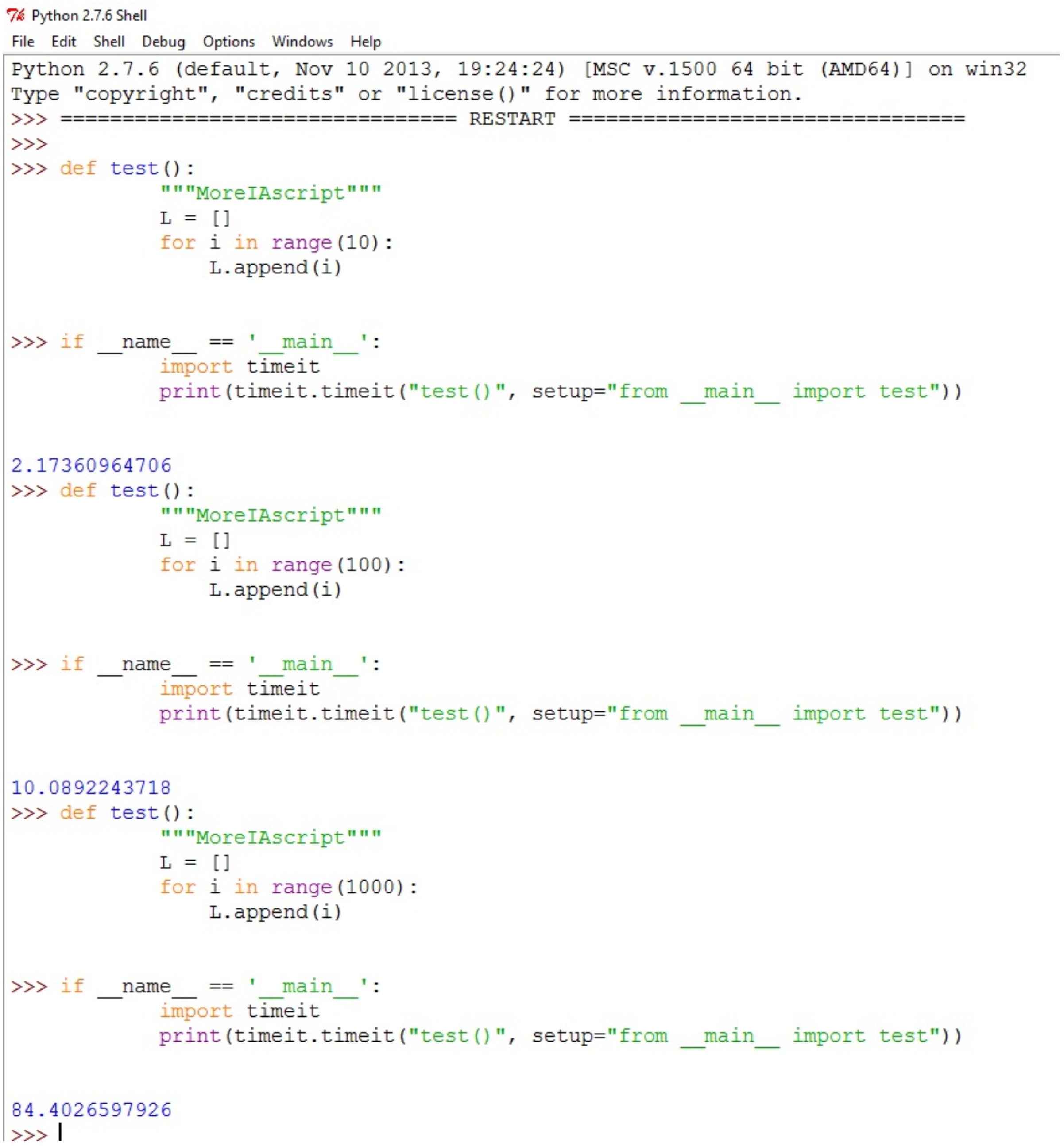


```
Python 2.7.6 (default, Nov 10 2013, 19:24:24) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Input  x  > y ,   a-y  > t ,   b-t  > h ,   t  > a-x ,
>>> totaltuple
defaultdict(None, {'a-h': ('x',), 'h': ('y',), 'b-h': ('x',), 'a-y': ('x',), 'a-x': ('x',), 't':
('x',), 'b-t': ('x',), 'y': ('x',), 'b-y': ('x',), 'a-t': ('x',)})
>>> countermodel('y','x')
defaultdict(None, {'a-h': set([]), 'h': set([]), 'b-h': set([]), 'a-y': set([1, 2, 3, 4, 5, 6, 7
, 'a-x', 'a-t']), 'a-x': set([1]), 't': set([1, 2, 3, 4, 'a-x']), 'b-t': set([1, 2, 3]), 'y': se
t([1, 2, 3, 4, 5, 6, 7, 'a-x', 'a-t']), 'x': set([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 'a-x', 'b-y', '
a-t']), 'b-x': set([1]), 'b-y': set([1]), 'a-t': set([1])})
>>> |
```

Figure 3.3: A counter-model example for Python implementation.

In Figure 3.3, inputs of the set of sentences are expressed as $'x > y, a-y > t, b-t > h, t > a-x'$ in abbreviation. $x > y$ means $'morexthany'$. $a-y$ contains a $-$ sign to indicate that a is an intersecting adjective and y is a plural noun. All the derivations to be obtained from this input set are indicated by *totaltuple*. $'A-h:(x)'$ means "*x more than a-h*" . The function $counter-model('y','x')$, which brings the counter-model to the square, checks whether the expression "*y more than x*" is derived first. If such a sentence is derived it prints '*derivable*'. Since this sentence can not be derived as in the example, we are calculating an counter-model. An empty set (set ([])) is assigned to the '*h*' which is not greater than any nouns in the set. Since '*h*' is an empty set, naturally 'b-h' appears to be empty. On the other hand, the set 'x' contains $0,1,2,3,4,5,6,7,8,9,10$ elements since it has cardinalities larger than 11 . The sets '$a-x$', '$b-y$', and '$a-t$' are contained as elements by the set 'x' because they have different sets and a smaller cardinality of 'x'. Otherwise, in a possible case, '$a-x$' and 'x' have the same number of elements and falsify the expression $('a-x':(x))$ in the set. To overcome this contradiction, we added cardinality to the 'x' cluster as a small 'a-x' expression element. Thus, the obtained model falsifies the expression 'x', but all the expressions of the input (including derivations) are confirmed. The algorithm type used here is based on topological ordering. When topological and cardinality comparisons are made, a comparison of the first and the following (larger or smaller cardinal quality) nouns are provided.

Figure 3.4 shows the results for the number of inputs in a computer with a 64-bit, 2.40 GHz, 8.00 GB RAM processor system. Based on the measurements it is seen that the algorithms have an effective time. Approximately $O(n) \cong \frac{n}{10}$ has the value of big O.



```
Python 2.7.6 (default, Nov 10 2013, 19:24:24) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
>>> def test():
            """MoreIAscript"""
            L = []
            for i in range(10):
                L.append(i)


>>> if __name__ == '__main__':
            import timeit
            print(timeit.timeit("test()", setup="from __main__ import test"))


2.17360964706
>>> def test():
            """MoreIAscript"""
            L = []
            for i in range(100):
                L.append(i)


>>> if __name__ == '__main__':
            import timeit
            print(timeit.timeit("test()", setup="from __main__ import test"))


10.0892243718
>>> def test():
            """MoreIAscript"""
            L = []
            for i in range(1000):
                L.append(i)


>>> if __name__ == '__main__':
            import timeit
            print(timeit.timeit("test()", setup="from __main__ import test"))


84.4026597926
>>> |
```

Figure 3.4: Input-output time measurements.

4. Conclusion

In this paper, for the purpose of comparing the sentences composed of cardinality comparisons of the expressions including the plural nouns and the intersecting adjectives have introduced and identified for the first time, and the derivation and counter model algorithms belonging to $L(More,IA)$ are given. An application of these algorithms with effective time complexity is built using a Python program.Nouns of the logic could be extended by adding their noun-level complements (such as non-cats, non-cats). The models of the logic are built on the finite sets. It can operate on Π^0_1 classes for models that can be built on infinitely countable clusters [9, 10, 11]. The paper [4] contains infinite models can be used to work on infinite models.

Acknowledgement

We would like to thanks anonymous referees and editor of the journal.

References

[1] Łukasiewicz, J., *Aristotle's syllogistic from the standpoint of modern formal logic*, Oxford University Press Academic Monograph Reprints, 222p., 1957.
[2] Moss, L.S., *Intersecting adjectives in syllogistic logic*, In The Mathematics of Language (pp. 223-237), Springer Berlin Heidelberg, 2010.
[3] Moss, L.S., *Syllogistic Logic with Cardinality Comparisons*, In J. Michael Dunn on Information Based Logics (pp. 391-415). Springer International Publishing, 2016.

[4] Moss, L.S. and Topal, S. *Syllogistic Logic with Cardinality Comparisons, On Infinite Sets*, arXiv preprint arXiv:1705.03037, (under review in RSL), 2017.

[5] Topal, S., *Bazı Sillojistik ve Kardinalite Karşılaştırmalı Lojiklerin Türetimlerinin Cebirsel ve Etiketli Çizge Teorik Özellikleri Üzerine*, Cilt 21, Sayı 3, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, DOI-10. 19113/sdufbed.50072, 2017.

[6] West, D.B., *Introduction to graph theory (Vol. 2)*, Upper Saddle River: Prentice hall, 2001.

[7] van Rossum, G., *Python Programming Language*, In USENIX Annual Technical Conference (Vol. 41, p. 36), 2017.

[8] Topal, S. and Akünsoy Y., *L(More, IA) Lojiğinin Python Uygulaması*, http://pbs.beu.edu.tr/s.topal/docs (Access date: 15.05.2017).

[9] Çevik, A., *Hesaplanabilirlik kuramı ve Turing derecelerine giris*, Gaziosmanpaşa Bilimsel Araştırma Dergisi, sayı 1, sayfa 1-20, 2012.

[10] Çevik, A., *Antibasis theorems for classes and the jump hierarchy*, Archive for Mathematical Logic, 137-142, 2013.

[11] Çevik, A., *choice classes*, Mathematical Logic Quarterly, 62(6), pp.563-574, 2016.