Programming with Term Logic

J. Martín Castro-Manzano
Faculty of Philosophy & Humanities UPAEP
Av 9 Pte 1908, Barrio de Santiago, 72410 Puebla, Pue., Mexico
+52 (222) 229 94 00
josemartin.castro@upaep.mx

L. Ignacio Lozano-Cobos
Faculty of Philosophy & Humanities UPAEP
Av 9 Pte 1908, Barrio de Santiago, 72410 Puebla, Pue., Mexico
+52 (222) 229 94 00
luisignacio.cobos@upaep.edu.mx

Paniel O. Reyes-Cárdenas
Faculty of Philosophy & Humanities UPAEP
Av 9 Pte 1908, Barrio de Santiago, 72410 Puebla, Pue., Mexico
+52 (222) 229 94 00
panielosberto.reyes@upaep.mx

Abstract:

Given the core tenets of Term Functor Logic and Aristotelian Databases, in this contribution we present the current advances of a novel logic programming language we call Term Functor Logic Programming Language.

Keywords: Syllogistic, logic programming, Aristotelian Database.

1. Introduction

Under direct influence of (Sommers, 1967 ; Sommers, 1982 ; Sommers & Englebretsen, 2000, Thompson, 1982 ; Mostowski, 1957), in other place we have offered an intermediate term logic for relational syllogistic that is able to deal with a wide range of common sense inference patterns (we call it the system TFL^+); an under direct influence of (Englebretsen, 1987 ; Englebretsen, 1991 ; Englebretsen, 1996 ; Englebretsen & Sayward, 2011) and TFL^+, in other place we have presented the diagrammatic counterpart of TFL^+ as to perform visual reasoning (we call it the system TFL⊕) (Castro-Manzano & Pacheco-Montes, 2018). Now, under the influence of this couple of systems and the notion of Aristotelian Database (Mozes, 1989), here we introduce the current advances of a novel logic programming language we call *Term Functor Logic Programming Language* (TFL^{PL}).

To better understand these advances consider, as in the following diagram, that the language we are suggesting here is the result of a series of works whose root lies in the logic of Aristotle and go down three different paths: one for Sommers' algebraic term logic, one for Leibniz's linear diagrams, and one for Veatch's realist interpretation of Aristotelian logic. These paths are grounded within three specific implementations: Sommers' logic, together with the recognition of the importance of non-classical quantifiers, result in the system TFL^+; Sommer's project and Leibniz's import on terms and linear diagrams impact on Englebretsen's systems that, in turn, influence the system TFL⊕; and Veatch's ideas influence the computational work developed by Mozes. Together, these last three implementations guide the maturation of TFL^{PL} so that we obtain a computational application defined after an enriched logic of terms.

$$\begin{tikzcd}
 & & & Aristotle(1989) \arrow[dl] \arrow[d] \arrow[dr] & \\
 & Mostowski(1957) \arrow[dr] & Sommers(1982) \arrow[d] \arrow[dr] & Leibniz(1961) \arrow[l] \arrow[d] & Veatch(1970) \arrow[d] \\
Peterson(1979) \arrow[r] & Thompson(1982) \arrow[r] & \mathsf{TFL}^{+} \arrow[d] \arrow[dr] & Englebretsen(1996) \arrow[dl] & Mozes(1989) \arrow[dl] \\
 & & \mathsf{TFL}^{\oplus} \arrow[r] & \mathsf{TFL}^{\mathsf{PL}} &
\end{tikzcd}$$

Hence, in this contribution we try to reach a simple goal: to present the advances of a programming language designed after the system TFL^{+}. In order to reach this goal we briefly present the system TFL (Sommers 1967, Sommers 1982, Sommers and Englebretsen 2000, Englebretsen 1987, Englebretsen 1996, Englebretsen and Sayward 2011) in the context of what we have called «the received view of logic» (§2) and the concept of Aristotelian Database (§3); then we expound the advances of the programming language TFL^{PL} by giving some details about its syntax and its working based upon the system TFL^{+} (§4); finally, we close with some remarks concerning future work (§5).

2. The received view of logic

Logic is about inference and in order to study inference we usually make use of first order languages. Thus, for example, propositional logic, first order logic, and first order logic with identity are logical systems defined by first order languages: $\{p, q, r, ..., \neg, \Rightarrow\}$, $\{a, b, c, ..., x, y, z, ..., f, g, h, ..., A, B, C, ..., \neg, \Rightarrow, \forall, \exists\}$, and $\{a, b, c, ..., x, y, z, ..., f, g, h, ..., A, B, C, ..., \neg, \Rightarrow, \forall, \exists, =\}$, respectively. The origin of this habit is associated with the representative advantages first order languages provide when compared to traditional systems. Russell (1937), for instance, made popular the idea that the limits of the traditional logic programme, i.e. syllogistic (*vide* Appendix A), were due to a commitment to a syntax of terms, that is, a grammar of triads composed by a subject term and a predicate term joined by a copula. Carnap (1930) generalized this idea to all traditional logic when he maintained that its available syntax was predicative only, as in "All (some) Greeks are (not) mortal" or "Socrates is (not) mortal".

Although it is true that the syntactical shortcomings of the ternary syntax (subject-copula-predicate) produce difficulties for the clear representation of singular, relational, or compound propositions, the major problem ternary syntax creates is term homogeneity. Geach argues:

> Our distinction between names and predicables enables us to clear up the confusion, going right back to Aristotle, as to whether there are genuine negative terms: predicables come in contradictory pairs, but names do not, and if names and predicables are both called "terms" there will be a natural hesitation over the question "Are there negative terms?" (Geach, 1980, p. 64)

According to this argument, term homogeneity does not allow us to preserve the fundamental noun-verb distinction. This incapacity of the ternary syntax is problematic because the function of a noun and the function of a predicate are not interchangeable: while the function of a noun is naming, the function of a predicate is predicating. Thus, as Geach argues, the interchange of subject and predicate terms is a syntactical issue that creates an undesired semantic effect, for only a noun can be a logical subject, but a noun cannot maintain its role as a noun if it suddenly becomes a predicate. So, this syntactical issue turns out to be also a semantic impossibility: between a term logic and genuine logic, goes Geach (1962, p. 54), there can only be war.

By contrast, genuine logic, namely first order logic (FOL), follows the syntax of the Fregean paradigm that results from dropping terms and favoring a binary grammar of function-argument pairs. These pairs promote a syntax that includes individual constants (a, b, c, ...) or variables (x, y, *z*, ...) as arguments that refer to individual objects as logical subjects, plus relations (*A*, *B*, *C*, ...) as functions that refer to concepts, not objects, as logical predicates. Thus, for instance, a proposition like "Socrates is mortal" could not be understood as a relation between a subject term and a predicate term, but as a function-argument pair were a constant, a saturated and complete element, say *s*, denotes an object named "Socrates" that works as an argument of the unsaturated and incomplete expression "… is mortal", say *Mx*, in such a way that *Ms* represents the proposition "Socrates is mortal": clearly, this syntactical representation does not allow any term shifting. Moreover, given this binary syntax, a proposition like "All men are mortal" cannot be understood as a relation of terms but as a relation between variables and quantifiers, say $\forall x(Hx \Rightarrow Mx)$, so that there are clear syntactic differences between singular propositions (like "Socrates is mortal") and universal propositions (like "All men are mortal"): this will be relevant below.

Hence, according to these remarks, genuine logic follows the syntax of the Fregean paradigm that results from dropping the use of a ternary syntax (subject-copula-predicate) in order to favor a binary syntax (function-argument). This binary syntax promotes the use of first order languages. This syntactical choice is the one that seems familiar to us because it is the one we follow when we teach, research, and apply contemporary logic: this is the received view of logic. However, it comes as no surprise that this view might very well feel familiar, but it is certainty not natural. Woods comments (emphasis is ours):

> *It is no secret that classical logic and its mainstream variants aren't much good for human inference as it actually plays out in the conditions of real life*—in life on the ground, so to speak. It isn't surprising. *Human reasoning is not what the modern orthodox logics were meant for*. The logics of Frege and Whitehead & Russell were purpose-built for the pacification of philosophical perturbation in the foundations of mathematics, notably but not limited to the troubles occasioned by the paradox of sets in their application to transfinite arithmetic. (Woods, 2016, p. 404)

So, even if genuine logic (classical, according to Woods) has been fundamental for the study of inference, both in cognitive science and artificial intelligence, it amazes us that, despite its original purpose, it is constantly used as a *bona fide* tool for representing natural language reasoning. Let us consider, to this effect, what we call "Bar-Hillel's challenge":

> I evaluated as to its validity with the help of formal logic. I regard this fact as one of the greatest scandals of human existence. Why has this happened? How did it come to be that logic which, at least in the views of some people 2,300 years ago, was supposed to deal with evaluation of argumentation in natural languages, has done a lot of extremely interesting and important things, but not this? (Staal, 1969, p. 256)

This is certainly a scandal. However, since the late 60's Fred Sommers defended a revision of the traditional ternary syntax under the veil of Bar-Hillel's challenge. Sommers, closer to a project of naturalization of logic, was concerned with how we reason. So, for instance, Sommers asked himself how is that a rational agent realizes that the following pair of beliefs is inconsistent:

(C_1) All dogs are animals but (C_2) someone who loves a dog does not love an animal.

Of course FOL is capable of offering an answer to the previous problem by way of first order languages (let C_1 be $\forall x(Px \Rightarrow Ax)$; and C_2 be $\exists x \exists y((Py \land Qxy) \land \forall z(Az \Rightarrow \neg Qxz)))$ and the proper deductive methods (Table 1), but as we will see, this capacity is not necessarily relevant for understanding natural language reasoning.

Table 1. A proof of the contradiction between C_1 and C_2 in FOL

1	∀x(Px⇒Ax)	C_1
2	∃x∃y((Py∧Qxy)∧∀z(Az⇒¬ Qxz))	C_2
3	∃y((Py∧Qay)∧∀z(Az⇒¬Qa z))	E∃ 2, a/x
4	(Pb∧Qab)∧∀z(Az⇒¬Qaz)	E∃ 3, b/y
5	Pb∧Qab	E∧ 4
6	∀z(Az⇒¬Qaz)	E∧ 4
7	Ab⇒¬Qab)	E∀ 6, b/z
8	Pb⇒¬Ab)	E∀ 1, b/x
9	Pb	E∧ 5
10	Qab	E∧ 5
11	Ab	E⇒ 8 and 9
12	¬Qab	E⇒ 7 and 11
13	Qab∧¬Qab	I∧ 10
14	X	EFSQ 13

3. Basic Aspects of Term Functor Logic

In this context, Sommers (1967, 1982, 2000) and Englebretsen (2000, 1987, 1996) developed a plus-minus algebra, *Term Functor Logic* (TFL), that deals with syllogistic by using terms rather than first order language elements such as individual variables or quantifiers.[10] According to this algebra, the four categorical propositions can be represented by the following syntax:[11]

- SaP := -S+P = -S-(-P) = -(-P)-S = -(-P)-(+S)
- SeP := -S-P = -S-(+P) = -P-S = -P-(+S)
- SiP := +S+P = +S-(-P) = +P+S = +P-(-S)
- SoP := +S-P = +S-(+P) = +(-P)+S = +(-P)-(-S)

Given this algebraic representation, the plus-minus algebra offers a sound, complete, and simple method of decision for syllogistic: a conclusion follows validly from a set of premises if and only if *i)* the sum of the premises is algebraically equal to the conclusion and *ii)* the number of conclusions with particular quantity (viz., zero or one) is the same as the number of premises with particular quantity (Englebretsen, 1996, p. 167). Thus, for instance, if we consider a valid syllogism from figure 1, we can see how the application of this method produces the right conclusion (Table 2).

[10]That we can reason without first order language elements such as individual variables or quantifiers is not news (cf. Quine 1971, Noah 1980, Kuhn 1983), but Sommers' logical project has a wider impact: that we can use a logic of terms instead of a first order system has nothing to do with the mere syntactical fact, as it were, that we can reason without quantifiers or variables, but with the the general view that natural language is a source of natural logic (cf. Sommers 1982, Sommers 2005, Moss 2015).

[11]We mainly focus on the presentation by Englebretsen (1996).

Table 2. An aaa-1 type syllogism

	Proposition	Representation
1.	All dogs are animals.	-D+A
2.	All German Shepherds are dogs.	-G+D
⊢	All German Shepherds are animals.	-G+A

In the previous example we can clearly see how the method works: *i)* if we add up the premises we obtain the algebraic expression (-D+A)+(-G+D)=-D+A-G+D=-G+A, so that the sum of the premises is algebraically equal to the conclusion and the conclusion is -G+A, rather than +A-G, because *ii)* the number of conclusions with particular quantity (zero in this case) is the same as the number of premises with particular quantity (zero).

This algebraic approach is also capable of representing relational, singular, and compound propositions with ease and clarity while preserving its main idea, namely, that inference is a logical procedure between terms. For example, the following cases illustrate how to represent and perform inferences with relational (Table 3), singular[12] (Table 4), or compound propositions[13] (Table 5). For a brief but systematic explanation of the rules employed in what follows *vide* Appendix B.

Table 3. A reasoning with relational propositions

	Proposition	Representation	Rule
1.	Some horses are faster than some dogs.	+H+(+F+D)	P
2.	Dogs are faster than some men.	-D+(+F+M)	P
3.	That which is faster than what is faster than some men, is faster than some men.[14]	-(+F+(+F+M))+(+F+M)	P
4.		+H+(+F+(+F+M))	DON 1,2
⊢	Some horses are faster than some men.	+H+(+F+M)	DON 3,4

Table 4. A reasoning with singular propositions

	Proposition	Representation	Rule
1.	All men are mortal.	-M+L	P
2.	Socrates is a man.	-s+M	P
⊢	Socrates is mortal.	-s+L	DON 1,2

Table 5. A reasoning with compound propositions

	Proposition	Representation	Rule
1.	If *P* then *Q*	-[p]+[q]	P
2.	*P*	+[p]	P
⊢	*Q*	+[q]	DON 1,2

These basic notions of TFL are sufficient for offering a solution to the problems posed by traditional term logic that we metioned above. But moreover, these features provide elements that

[12]Provided singular terms, such as *Socrates*, are represented by lowercase letters.

[13]Given that compound propositions can be represented as follows, *P*:=`[p]`, *Q*:=`[q]`, ¬*P*:=`-[p]`, *P*→*Q*:=`-[p]+[q]`, *P*∧*Q*:=`+[p]+[q]`, and *P*∨*Q*:=`--[p]--[q]`, the method of decision behaves like resolution (cf. Noah 2005).

[14]Or in other words: the relation *faster than* is transitive.

allow us to explain, for instance, why we instantaneously judge that (C_1) *All dogs are animals* and (C_2) *Someone who loves a dog does not love an animal* are mutually inconsistent. Let us consider, to this effect, the proof of the contradiction between C_1 and C_2 in TFL (Table 6).

Table 6. A proof of the contradiction between C_1 and C_2 in TFL

1	$-P_2+A_2$	C_1
2	$+(+Q_{12}+P_2)-(+Q_{12}+A_2)$	C_2
3	$+(+Q_{12}+A_2)-(+Q_{12}+A_2)$	DON 1, 2

If we compare the proof of the contradiction between C_1 and C_2 in FOL (Table 1) with the proof in TFL (Table 6), we can clearly observe that in FOL not only we need a more complex representation by using first order elements (say, variables and quantifiers), but also a more complex proof in order to justify there is a contradiction, and yet, such proof does not provide any lights as to why we instantaneously see a contradiction between C_1 and C_2. By contrast, TFL has a more natural grammar and a simpler set of rules for performing inference. And since FOL, due to its very origin, does not have these features, it cannot offer cognitively relevant information as to how we actually reason. And so, even if FOL has merits within the foundations of mathematics, it cannot be the logic of natural language. And as we can suspect at this point, this story is akin to the history of logic programming languages.

4. Aristotelian Databases

Indeed, according to Mozes (1989), the abandon of traditional logic, i.e. syllogistic, by computer scientists in order to favor FOL is not justified (cf. Kowalski, 1988). Syllogistic is a term logic that was originally created with the purpose of understanding and guiding human reasoning; but as we have seen, this was not the original purpose of FOL, and so a term logic not only has cognitive import, but also computational interest. Consequently, Mozes developed the concept of *Aristotelian Database* based upon traditional logic.

A database is Aristotelian when it has the following features:

- The ability to give natural-language explanations of deductions.
- The ability to volunteer information, in answer to yes/no questions, if a stronger or weaker version of the "yes" answer can be proved.
- The ability to point out results that cannot be proved but seem to be likely possibilities.
- The ability, in answer to yes/no questions, to suggest "missing rules", i.e. rules that if added to the database, will allow proving a "yes" answer.
- The ability to suggest instances in which non-deductive forms of reasoning, such as analogy or induction, are likely to be useful.

To obtain these features, the structure of a database *à la* Mozes is defined by a set of constants that represent objects and a set of relations that represent properties (we will refer to this issue later). Information about objects is expressed by facts, that is, by relations applied to objects, for instance, man (Socrates). In this sense, these databases follow a syntax similar to Prolog's (cf. Sterling & Shapiro, 1994; Bratko, 2001). A rule, on the other hand, consists of a subject that is the conjunction of one or more relations applied to variables or constants, and a predicate that is a unique relation; plus a type of rule that indicates the nexus between the subject and the predicate. When a rule is written we first write down the predicate, then the rule type, and finally the subject. There are four possible rule types that correspond to the four categorical propositions (*vide* Appendix A). Thus, for instance, mortal(X) A man(X) means *All men are mortal* (in Prolog, mortal(X):−man(X).).

In these databases there are two types of queries: answer queries that specify a fact or a rule to be proved; and retrieval queries that return an answer given the compliance of a conjunction of relations. After getting an answer, the user may ask for an explanation. If the query was a retrieval query, the user may specify a constant and ask why such a constant was not returned. If the query is an answer query and it was not proven as true or false, the user may ask for implicit rules. Besides the ability to prove negative facts, Aristotelian Databases also implement negation by failure. Finally, as can be noted, the deductive processes of these databases are based upon syllogistic.

Consider, as an example, the next fragment adapted from (Mozes, 1989):

```
\\Facts
man(Socrates)
wise(Socrates)
man(Joe)
age(Joe,1)
man(John)
age(John,15)
man(Peter)
age(Peter,40)
dog(Fido)
work_of_art(MonaLisa)

\\Rules
animal(X) A man(X)
animal(X) A dog(X)
mortal(X) A man(X)
mortal(X) E work_of_art(X)
man(X) E dog(X)
responsible(X) E ¬man(X)
responsible(X) A wise(X)
responsible(X) E baby(X)
responsible(X) I adult(X)
responsible(X) E idiot(X)
age(X,0-120) A man(X)
baby(X) A (man(X)^age(X,0-2))
adult(X) A (man(X)^age(X,21-120))
idiot(X) I (man(X)^age(X,30-50))

\\Example
> mortal(X) A ¬responsible(X)
Joe
By negation as failure: John
Possibility: Fido
> Explain Joe
Because Joe is a man, he is mortal.
Because Joe is a man with age 1, he is a baby.
Because Joe is a baby, he is not responsible.
> Explain Fido
Because Fido is a dog, it is an animal.
Some animal is mortal; for example, the men.
Because Fido is an animal, it is, perhaps, mortal.
Because Fido is a dog, it is not a man.
Because Fido is not a man, it is not responsible.
> Explain Socrates
Because Socrates is wise, he is responsible.
> Explain Peter
Because Peter is a man with age 40, he is an adult.
Because Peter is an adult, he is, perhaps, responsible.
```

Aristotelian Databases, then, use syllogisms as models of inference. The main advantage of this Aristotelian approach is its cognitive closeness with natural language reasoning. This suggests, in Mozes' opinion, two applications in which traditional logic could be useful: applications that require human interaction with natural language, and applications that try to emulate human reasoning.

5. An Introduction to Term Logic Programming

Now, if we consider the first application, we require a more powerful Aristotelian Database, namely, one capable of dealing with a wide range of common sense inference patterns but also cognitively closer to natural language. To do this we first introduce the basic elements of the system TFL^+ and then we present the advances of TFL^{PL} as a programming language designed after TFL^+.

Peterson (1979) and Thompson (1982) developed extensions for syllogistic ($SYLL^+$) by adding some extra quantifiers, namely, "most" (for majority propositions), "many" (for common propositions), and "few" (for predominant propositions).[15] So, this framework adds the next propositions: p is the predominant affirmative (*Few* S *are not* P), b is the predominant negative (*Few* S *are* P), t is the majority affirmative (*Most* S *are* P), d is the majority negative (*Most* S *are not* P), k is the common affirmative (*Many* S *are* P), and g is the common negative (*Many* S *are not* P). This framework allows us to extend syllogistic as to cope with a wide range of common sense reasoning patterns. As expected, the addition of p, t, k, b, d, and g increases the number of valid syllogistic moods (Table 7).

Table 7. Valid syllogistic moods according to $SYLL^+$ adapted from (Thompson 1982)

	Figure 1	Figure 2	Figure 3	Figure 4
With "most"	aat att ati ead etd eto	aed add ado ead etd eto	ati eto tai dao	aed eto tai
With "many"	aak atk aki akk eag etg eko ekg	aeg adg ago agg eag etg eko ekg	aki eko kai gao	aeg eko kai
With "few"	aap app apt apk api eab epb epd epg epo	aeb abb abd abg abo eab epb epd epg epo	pai epo bao api	aeb pai epo

The plus-minus algebra of TFL, as we have seen, provides a simple and logically sound algebraic approach for syllogistic that, alas, does not cover cases of common sense reasoning involving non-classical quantifiers such as "most," "many," or "few;" on the other hand, the syllogistic extended with extra quantifiers comprises a wide range of common sense inference patterns but, unfortunately, it lacks an algebraic procedure. So, given this state of affairs, TFL^+ is system of syllogistic that includes such a broad range of inferential patterns but with the virtues of an algebraic approach. In order to represent propositions p, t, k, b, d, and g within the framework of the plus-minus algebra, TFL^+ uses the proposal displayed in Table 8.

[15]We mainly focus on the presentation by Thompson (1982).

Table 8. Representation of the syllogistic propositions

Proposition		Representation	Proposition		Representation
SaP	:=	$-S^0+P^0$	SeP	:=	$-S^0-P^0$
SpP	:=	$+S^3+P^0$	SbP	:=	$+S^3-P^0$
StP	:=	$+S^2+P^0$	SdP	:=	$+S^2-P^0$
SkP	:=	$+S^1+P^0$	SgP	:=	$+S^1-P^0$
SiP	:=	$+S^0+P^0$	SoP	:=	$+S^0-P^0$

Given this new representation, the modification of the plus-minus algebra method of decision is as follows: a conclusion follows validly from a set of premises if and only if *i)* the sum of the premises is algebraically equal to the conclusion, *ii)* the number of conclusions with particular quantity is the same as the number of premises with particular quantity, and *iii)* the level of quantification of the conclusion is lesser or equal than the maximum level of quantification of the premises.

The performance of this tweaked version of syllogistic may be better appreciated by considering the trade-off between the complexity of the SYLL$^+$ framework and the expressive power of the TFL framework regarding common sense reasoning with non-classical quantifiers. To illustrate this, let us consider some examples (Tables 10-13). As expected, this tweaked version of syllogistic allows the valid inference patterns displayed in Table 9.[16]

Table 9. Valid syllogistic patterns with extra quantifiers in the TFL$^+$ framework

	Figure 1	Figure 2	Figure 3	Figure 4
With "most"	att ati etd eto	add ado etd eto	ati eto tai dao	eto tai
With "many"	atk aki akk etg eko ekg	adg ago agg etg eko ekg	aki eko kai gao	eko kai
With "few"	app apt apk api epb epd epg epo	abb abd abg abo epb epd epg epo	pai epo bao api	pai epo

Table 10. An invalid reasoning: kaa-1

	Proposition	Representation
1.	Many homeless are ill.	$+H^3+I^0$
2.	This guy is homeless.	$-g^0+H^0$
⊭	This guy is ill.	$-g^0+I^0$

[16]For the valid inferential patterns that need existential import, like `aat`-1 or `aak`-1, the only requirement is to add the missing implicit premise that states the existence of the minor term, namely, something akin to $+S^0+S^0$.

Table 11. An invalid reasoning: akt-4

	Proposition	Representation
1.	All cops are fascists.	$-C^0+F^0$
2.	Many men are cops.	$+M^1+C^0$
$\nvdash$	Most men are fascists.	$+M^2+F^0$

Table 12. A valid reasoning: bao-3

	Proposition	Representation
1.	Few cars are hybrid.	$+C^3-H^0$
2.	All cars are expensive.	$-C^0+E^0$
$\vdash$	Some expensive cars are not hybrid.	$+E^0-H^0$

Table 13. A valid reasoning: ekg-2

	Proposition	Representation
1.	No fool is a citizen.	$-F^0-C^0$
2.	Most voters are citizens.	$+V^2+C^0$
$\vdash$	Many voters are not fools.	$+V^1-F^0$

As we can see from these examples, the TFL$^+$ framework gains the advantages of an algebraic method (a reduction of a complex set of rules into a simple and unified formal approach) and, at the same time, it gains the advantages of a theory of syllogisms with non-classical quantifiers (an assessment of a wide range of common sense inference patterns that extends the scope of traditional syllogistic), with the addition that the TFL$^+$ framework is reliable in that all valid syllogisms in the SYLL$^+$ framework can be obtained by applying the modified plus-minus algebra method of decision, and vice versa, all syllogisms that can be obtained by applying the modified plus-minus algebra method of decision are valid syllogisms in the SYLL$^+$ framework:

Proposition 1 (Reliability) An inference is SYLL$^+_{\text{valid}}$ iff it is TFL$^+_{\text{valid}}$.

So, at this point we can say that, if the relevance of the reliability of TFL$^+$ has to do with expressive and algebraic limitations of FOL and TFL$^+$ and SYLL$^+$ with respect to common sense inference in natural language, the usefulness of Term Functor Logic Programming Language (TFL^{PL}) results from its potential use for systems of information retrieval in which natural language interaction is basic or fundamental. Given these remarks, TFL^{PL} is a language that, like Prolog, has a special grammar. The next fragment is an example of a program of TFL^{PL}:

```
-s0+H0
-H0+M0
```

The first line can be read as "Socrates is a man" (i.e., $-s^0+H^0$ in TFL$^+$; man(Socrates) in Prolog) while the second line represents "All men are mortal" (i.e., $-H^0+M^0$ in TFL$^+$; mortal(X):-man(X) in Prolog). This allows us to see that, just as in TFL the distinction between singular and universal proposition disappears, in TFL^{PL} the distinction between facts and rules also disappears: this is an important difference from other logic programming languages like Prolog; and this is also different from the Aristotelian Database originally proposed by Mozes (Table 14).

Table 14. Comparison

Proposition	FOL	TFL$^+$	Prolog	TFL^{PL}
Socrates is a man.	Ms	$-s^0+M^0$	man(Socrates).	-s0+M0
All men are mortal.	$\forall x(Hx \Rightarrow Mx)$	$-H^0+M^0$	mortal(X):-man(X).	-H0+M0
Many humans are Greek.	--	$+H^2+G^0$	--	+H2+G0

Given the previous program, we can make some inferential queries:

```
> s

-H0+M0
-s0+H0
------
-s0+M0
```

that is, "What is Socrates?" (s), and the program answers -s0+M0, that is to say, "Socrates is mortal." We can clearly see another difference between TFL^{PL} and Prolog. The syntax of TFL^{PL} is ternary, Aristolelian as it were, and thus, it organically induces a database *à la* Mozes; however, it different from the original notion of Aristotelian Database in another respect: TFL^{PL} avoids the choice of a Fregean binary syntax in order to favor a syntax closer to Sommers'. This results in a logic programming language based upon a system of natural logic rather than on first order logic.

As a result of being grounded on TFL$^+$, the syntax of TFL^{PL} is the same as that of TFL$^+$ with the restriction that the super index notation is not maintained: the super index is written immediately after the term. The syntax, then, can be summarized by the following BNF:

```
<program> ::= <proposition> | <proposition><program>
<proposition> ::= <term^n><±term0>
<term^n> ::= <±T0>|<+T1>|<+T2>|<+T3>
```

That is to say, a program in TFL^{PL} is a sequence of one or more propositions. And each proposition is defined by two terms as in TFL$^+$. This formal syntax allows us to define, in a rigorous way, the nexus between the philosophical and the cognitive motivation. Clearly, given this syntax, the formal semantics of TFL^{PL} is directly the semantics of TFL$^+$, and given Proposition 1, it is possible to claim TFL^{PL} preserves the technical results we pretend to capture, namely, the inferential process of a logic of terms.

The implementation of TFL^{PL} was achieved in C. At the moment, the inferential motor of TFL^{PL} receives a TFL^{PL} program and makes an iterative use of the inferential process defined previously, namely, that a conclusion follows validly from a set of premises if and only if *i)* the sum of the premises is algebraically equal to the conclusion, *ii)* the number of conclusions with particular quantity is the same as the number of premises with particular quantity, and *iii)* the level of quantification of the conclusion is lesser or equal than the maximum level of quantification of the premises.

As an example, consider the following program:

```
-s0+M0 // Socrates is a man.
-f0+D0 // Fido is a dog.
-M0+A0 // All men are animals.
-D0+A0 // All dogs are animals.
-M0+O0 // All men are mortal.
-A0+O0 // Few animals are not mortal.

> s // What is socrates.
```

```
-M0+A0 // All men are animal.
-s0+M0 // Socrates is a man.
-------
-s0+A0 // Therefore, Socrates is an animal.

-M0+O0 // All men are mortal.
-s0+M0 // Socrates is a man.
-------
-s0+O0 // Therefore, Socrates is mortal.
```

We can extract some remarks after the syntax, the implementation, and the example: *a)* since the syntax of TFL involves the abandon of the Fregean syntax, TFL^{PL} does not require the use of variables or constants. *b)* Since the syntax of TFL implies the rejection of the distinction between singular and universal propositions, the distinction between facts and rules disappears in TFL^{PL}. *c)* Given the previous remarks, TFL does need to use the equality predicate, =, and so in TFL^{PL} we have no need for substitution or unification procedures as in Prolog. *d)* TFL^{PL} allows us to define Aristotelian Databases in so far as it uses syllogistic inference and has a *leitmotiv* closer to natural language, even though TFL^{PL} has also a syntactic difference with Mozes' databases, namely, that TFL^{PL} has a ternary syntax. *e)* As of today, TFL^{PL} only uses propositions defined after pairs of terms, but since TFL is able to model more complex relations, we need to add a relational module to TFL^{PL}, as well as a module for numerical reasoning.

7. Final remarks

In this contribution we have tried to reach a simple goal: to present the advances of a programming language designed after the Intermediate Term Functor Logic. As of now, the implementation only uses propositions with two terms and the rule *DON*, but if we consider its working and its motivation, we can see that TFL^{PL} is a logic programming language that is quite promising, not only a computational device, but also as a research device associated to a cognitive project that will include inferential models for relational logic (in so far as TFL^{+} is capable of dealing with relational inferences), probabilistic reasoning (in so far as TFL^{+} accepts a probabilistic interpretation (cf. Thompson, 1986)), and numerical reasoning (in so far as we could incorporate numerical term logic (cf. Murphree, 1998)). Also, this programming language can be adapted into psychological accounts of common sense reasoning (cf. Keil, 2005 ; Khemlani & Johnson-Laird, 2012)) and philosophical discussions about the nature of inference (cf. Veatch, 1970 ; Sommers, 1982 ; Englebretsen, 1996, Englebretsen & Sayward, 2001)), not to mention it could have pedagogical advantages given its closeness to natural language reasoning.

References

Aristotle. (1989). *Prior Analytics*, Hackett Classics Series, Hackett.

Bratko, I. (2001). *Prolog Programming for Artificial Intelligence*, Addison Wesley.

Carnap, R. (1930). "Die alte und die neue logik", *Erkenntnis*, 1, pp. 12-26.

Castro-Manzano, J.M., & Pacheco-Montes, J.R. (2018). "Moded Diagrams for Moded Syllogisms", In Chapman P., Stapleton G., Moktefi A., Perez-Kriz S., Bellucci F. (eds.) *Diagrammatic Representation and Inference. Diagrams 2018*. Lecture Notes in Computer Science, vol 10871. Springer, Cham.

Couturat, L. (1961). *La logique de Leibniz d'après des documents inédits*, Hildesheim, G. Olms.

Englebretsen, G. (1987). *The New Syllogistic*, Peter Lang.

Englebretsen, G. (1991). "Linear Diagrams for Syllogisms (with Relationals)", *Notre Dame J. Formal Logic*, 33(1), pp.37-69.

Englebretsen, G. (1996). *Something to Reckon with: The Logic of Terms*, University of Ottawa Press.

Englebretsen, G., & Sayward, C. (2011). *Philosophical Logic: An Introduction to Advanced Topics*, Bloomsbury Academic.

Geach, P. (1962). *Reference and Generality: An Examination of Some Medieval and Modern Theories*, Cornell University Press.

Geach, P. (1980). *Logic Matters*, University of California Press.

Keil, F. (2005). “Exploring Boundary Conditions on the Structure of Knowledge: Some Nonobvious Influences of Philosophy on Psychology”, in David S. Oderberg (ed.), *The Old New Logic: Essays on the Philosophy of Fred Sommers*, Bradford, pp. 67-84.

Khemlani, S., & Johnson-Laird, P. N. (2012). “Theories of the Syllogism: a Meta-Analysis”, . *Psychological Bulletin*, pp. 427-457.

Kowalski, R. A. (1988). “The early years of logic programming,” *Commun. ACM*, 31(1), pp. 38-43.

Kuhn, S. (1983). “An Axiomatization of Predicate Functor Logic”, *Notre Dame J. Formal Logic*, 24(2), pp. 233-241.

Mostowski, A. (1957). “On a Generalization of Quantifiers”, *Fundamenta Mathematicae*, 44(2), pp. 12-36.

Moss, L. (2015). “Natural logic”, in S. Lappin y C. Fox (eds.), *The Handbook of Contemporary Semantic Theory*, John Wiley & Sons.

Mozes, E. (1989). “A Deductive Database Based on Aristotelian Logic”, *Journal of Symbolic Computation*, 7(5), pp. 487-507.

Murphree, W. (1998). “Numerical Term Logic”, *Notre Dame J. Formal Logic,*39(3), pp. 346-362.

Noah, A. (1980). “Predicate-functors and the Limits of Decidability in Logic”, *Notre Dame J. Formal Logic*, 21(4), pp. 701-707.

Noah, A. (2005). “Sommers’s Cancellation Technique and the Method of Resolution”, in David S. Oderberg (ed.), *The Old New Logic: Essays on the Philosophy of Fred Sommers*, Bradford, pp.169- 182.

Peterson, P. L. (1979). “On the Logic of “few”, “many”, and “most”,” *Notre Dame J. Formal Logic*, 20, pp. 155-179.

Quine, W. Van O. (1971). “Predicate Functor Logic”, in J. E. Fenstad (ed.) *Proceedings of the Second Scandinavian Logic Symposium*, North-Holland.

Russell, B. (1937). *A critical exposition of the philosophy of Leibniz: with an appendix of leading passages*, G. Allen & Unwin.

Sommers, F. (1967). “On a Fregean Dogma”, in I. Lakatos (ed.), *Problems in the Philosophy of Mathematics*, volumen 47 Studies in Logic and the Foundations of Mathematics, Elsevier, pp.47- 81.

Sommers, F. (1982). *The Logic of Natural Language*, Oxford University Press.

Sommers, F. (2005). “Intellectual Autobiography”, in David S. Oderberg (ed.), *The Old New Logic: Essays on the Philosophy of Fred Sommers*, Bradford, pp. 1-24.

Sommers, F., & Englebretsen, G. (2000). *An Invitation to Formal Reasoning: The Logic of Terms*, Ashgate.

Staal, J. F. (1969). “Formal Logic and Natural Languages (a Symposium)”, *Foundations of Language* 5(2), pp. 256-284

Sterling, L., & Ehud, S. (1994). *The Art of Prolog: Advanced Programming Techniques,* MIT Press.

Thompson, Bruce (1982), “Syllogisms using “few”, “many”, and “most”,” *Notre Dame J. Formal Logic*, 23(1), pp. 75-84.

Thompson, B. (1986). “Syllogisms with Statistical Quantifiers”, *Notre Dame J. Formal Logic*, 27(1), pp. 93-103.

Veatch, H. B. (1970). *Intentional Logic: A Logic Based on Philosophical Realism*, Archon Books.

Woods, J. (2016). “Logic Naturalized”, In J. Redmond, O. Pombo, and A. Nepomuceno-Fernández (eds.), *Epistemology, Knowledge and the Impact of Interaction*, Springer, pp. 403-432.

Appendix A. Syllogistic

Syllogistic (SYLL) is a term logic that has its origins in Aristotle’s *Prior Analytics* (1989) and deals with the consequence relation between categorical propositions. A *categorical proposition* is a proposition composed by two terms, a quantity, and a quality. The subject and the predicate of a proposition are called *terms*: the term-schema S denotes the subject term of the proposition and the term-schema P denotes the predicate. The *quantity* may be either universal (*All*) or particular (*Some*) and the *quality* may be either affirmative (*is*) or negative (*is not*). These categorical propositions are denoted by a *label*, either a (universal affirmative, SaP), e (universal negative, SeP), i (particular affirmative, SiP), or o (particular negative, SoP) that allows us to determine a sequence of three propositions called *mood*. A categorical syllogism, then, is a mood ordered in such a way that two propositions are premises and the last one is a conclusion. Within the premises there is a term that appears in both premises but not in the conclusion. This special term, usually denoted with the term-schema M, works as a link between the remaining terms and is known as the middle term. According to the position of this last term, four *figures* can be set up in order to encode the valid syllogistic moods or syllogistic patterns (Table A1).[17]

Table A1. Valid syllogisms

Figure 1	Figure 2	Figure 3	Figure 4
aaa	eae	iai	aee
eae	aee	aii	iai
aii	eio	oao	eio
eio	aoo	eio	

Appendix B. Rules of inference for TFL

For the purposes of this study, here we expound the rules of inference for TFL as they appear in (Englebretsen 1996).

1. Rules of immediate inference.
    1. Premise (P): Any premise or tautology can be entered as a line in proof. (Tautologies that repeat the corresponding conditional of the inference are excluded. The corresponding conditional of an inference is simply a conditional sentence whose antecedent is the conjunction of the premises and whose consequent is the conclusion.)
    2. Double Negation (DN): Pairs of unary minuses can be added or deleted from a formula (i.e., --X=X).
    3. External Negation (EN): An external unary minus can be distributed into or out of any phrase (i.e., -(±X±Y)=±X±Y).
    4. Internal Negation (IN): A negative qualifier can be distributed into or out of any predicate-term (i.e.,±X-(±Y)=±X+(±Y)).
    5. Commutation (Com): The binary plus is symmetric (i.e., +X+Y=+Y+X).
    6. Association (Assoc): The binary plus is associative (i.e., +X+(+Y+Z)=+(+X+Y)+Z).
    7. Contraposition (Contrap): The subject- and predicate-terms of a universal affirmation can be negated and can exchange places (i.e., -X+Y=-(-Y)+(-X)).
    8. Predicate Distribution (PD): A universal subject can be distributed into or out of a conjunctive predicate (i.e., -X+(+Y+Z)=+(-X+Y)+(-X+Z)) and a particular subject can be distributed into or out of a disjunctive predicate (i.e., +X + (-(-Y) - (-Z))=--(+X + Y)--(+X + Z)).
    9. Iteration (It): The conjunction of any term with itself is equivalent to that term (i.e., +X+X=X).

2. Rules of mediate inference.
    1. (DON): If a term, M, occurs universally quantified in a formula and either M occurs not universally quantified or its logical contrary occurs universally quantified in another

[17] For sake of brevity, but without loss of generality, here we omit the syllogisms that require existential import.

formula, deduce a new formula that is exactly like the second except that M has been replaced at least once by the first formula minus its universally quantified M.

2. Simplification (Simp): Either conjunct can be deduced from a conjunctive formula; from a particularly quantified formula with a conjunctive subject-term, deduce either the statement form of the subject-term or a new statement just like the original but without one of the conjuncts of the subject-term (i.e., from +(+X+Y)±Z deduce any of the following: +X+Y, +X±Z, or +Y±Z), and from a universally quantified formula with a conjunctive predicate-term deduce a new statement just like the original but without one of the conjuncts of the predicate-term (i.e., from -X±(+Y+Z) deduce either -X±Y or -X±Z).
3. Addition (Add): Any two previous formulae in a sequence can be conjoined to yield a new formula, and from any pair of previous formulae that are both universal affirmations and share a common subject-term a new formula can be derived that is a universal affirmation, has the subject-term of the previous formulae, and has the conjunction of the predicate-terms of the previous formulae as its predicate-term (i.e., from -X+Y and -X+Z deduce -X+(+Y+Z>).

Table B1. An example of DON

	Proposition	Representation	Rule
1.	Every boy loves some girl.	$-B_1+(+L_{12}+G_2)$	P
2.	Every girl adores some cat.	$-G_2+(+A_{23}+C_3)$	P
□ □	All cats are mangy.	-C+M	P
□ □	Whatever adores something mangy is a fool.	$-(+A_{23}+M_3)+F$	P
□ □		$-G_2+(+A_{23}+M_3)$	2, 3, DON
□ □		-G+F	4, 5, DON
⊢	Every boy loves a fool	$-B_1+(+L_{12}+F_2)$	1, 6, DON