# Meaning Postulates in a Lexico-Conceptual Knowledge Base

Carlos Periñán Pascual
Unidad Central de Idiomas
Universidad Católica San Antonio de Murcia

Francisco Arcas Túnez
Departamento de Informática de Sistemas
Universidad Católica San Antonio de Murcia

**Abstract**

*Some natural language processing systems, e.g. machine translation, require to have a knowledge base with conceptual representations reflecting the structure of human beings' cognitive system. In some other systems, e.g. automatic indexing or information extraction, surface semantics could be sufficient, but the construction of a robust knowledge base guarantees its use in most natural language processing tasks, consolidating thus the concept of resource reuse. This paper describes how meaning postulates can be built through FunGramKB, resulting in a knowledge base which provides deep semantics.*

## 1: FunGramKB

FunGramKB is a lexicographical tool for the semiautomatic construction of a multipurpose lexico-conceptual knowledge base for a natural language processing (NLP) system within the theoretical model of S.C. Dik's Functional Grammar [1]. FunGramKB is not a literal implementation of Dik's lexicon, but we depart from the functional model in some important aspects with the aim of building a more robust knowledge base.

The objective of this paper is to describe meaning postulates that can be built through FunGramKB. Due to our conceptualist approach, firstly it is necessary to describe the way our ontological component is organized.

## 2: Ontological model in FunGramKB

In the field of knowledge engineering, the ontology is intended to store specific domain knowledge shared by a community. Particularly, FunGramKB's ontological component takes the form of a universal concept taxonomy, where 'universal' means that every concept we could imagine has an appropriate place in this ontology. Moreover, our ontology is linguistically motivated, as a result of its involvement with the semantics of lexical units, but the knowledge stored in our ontology is not specific to any particular language.

Our ontological model distinguishes three different conceptual levels, each one of them with concepts of a different type: metaconcepts, basic concepts and terminals (figure 1).

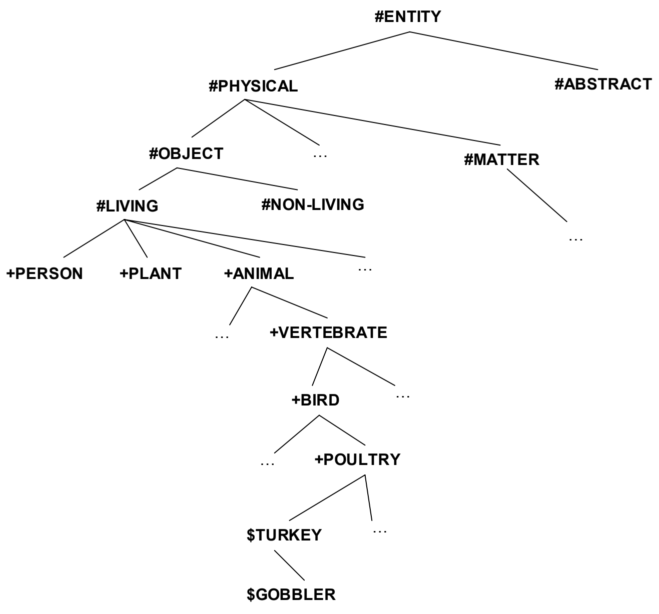


**Figure 1. An example of ontological structuring in FunGramKB.**

Metaconcepts, preceded by symbol #, constitute the upper level in the taxonomy. The analysis of the main upper-level linguistic ontologies led to a metaconceptual model whose design contributes to the integration and exchange of information with other ontologies. Moreover, our metaconcepts play the role of 'hidden categories', that is, concepts which aren't linked to any lexical unit so that they can serve as 'hidden' superordinates and avoid circularity. Root metaconcepts are *#ENTITY*, *#ATTRIBUTE* and *#EVENT*. One of the metaproperties of events is the prototypical thematic pattern, which is non-monotonically inherited as a conceptual frame by subordinate basic concepts and terminals, and finally linked to lexical units with the aim of building their frames.

Basic concepts, preceded by symbol +, are used in FunGramKB as defining units enabling the construction of meaning postulates for basic concepts and terminals, as well as taking part as selection preferences in conceptual and lexical frames.

The terminal level, where concepts are headed by symbol $, is not hierarchical structured; however, terminals are not presented indiscriminately as members of concept lists associated to basic concepts, but terminals are supplied with a series of properties. The border line between basic concepts and terminals is based on their definitory potential to take part in meaning postulates.

As far as basic concepts and terminals are concerned, properties of particular interest are the conceptual frame and the meaning postulate. As an example, the values of these properties for concept +FORBID are presented:

Conceptual frame
+FORBID $(x_1$: +PERSON$)_{Agent}$ $(x_2)_{Theme}$ $(x_3$: +PERSON$)_{Goal}$

Meaning postulate
+$(e_1$: +SAY $(x_1)_{Agent}$ $(x_3)_{Goal}$ $(x_4$: $(e_2$: $_n$ +DO $(x_3)_{Agent}$ $(x_2)_{Theme}))_{Theme})$

The interlinguistic nature of conceptual frames and meaning postulates is manifest in that these constructs are similar in the representation of the same state of affairs in different languages.

## 3: Meaning postulates in FunGramKB

### 3.1: Overview

Velardi *et alii* [8] distinguish two well-defined strategies when describing meaning in computational lexicography: i.e. the cognitive content in a lexical unit can be described by means of semantic features or primitives (conceptual meaning), or through associations with other lexical units in the lexicon (relational meaning). Strictly speaking, the latter doesn't give a real definition of the lexical unit, but it describes its usage in the language via 'meaning relations' with other lexical units. It is certainly easier to state associations among lexical units in the way of meaning relations than formally describing the cognitive content of lexical units, but the inference power of conceptual meaning is stronger. Surface semantics can be sufficient in some NLP systems, but the construction of a robust knowledge base guarantees its use in most NLP tasks, consolidating thus the concept of resource reuse.

In FunGramKB, the meaning postulate is conceived as a property of basic concepts and terminals. Root basic concepts are an exceptional case, because they are treated as semantic primitives. Current lexicalist models agree to handle lexical meaning as a cognitive representation reflecting the speakers' shared knowledge about the referent linked to a given linguistic expression. Therefore, when representing one of the meanings of a lexical unit, we are really representing the meaning of a concept. This is the reason why meaning postulates are processed as a conceptual property in FunGramKB.

### 3.2: Formal grammar of meaning postulates

In FunGramKB, a meaning postulate is a set of one or more logically connected predications, which are cognitive constructs carrying the generic features of the concept. In this section we describe the formal grammar which allows the machine to recognize well-formed predications for a meaning postulate. To illustrate, some predications in the meaning postulate of concept +BIRD are presented:

[a] +$(e_1$: +BE $(_1$ $x_1$: +BIRD$)_{Theme}$ $(_1$ $x_2$: +VERTEBRATE$)_{Referent})$
[b] *$(e_2$: HAVE $(_1$ $x_1)_{Theme}$ $(_1$ $x_3$: +FEATHER$)_{Referent}$ $(_2$ $x_3$: +LEG$)_{Referent}$ $(_2$ $x_3$: +WING$)_{Referent})$
[c] *$(e_3$: +FLY $(_1$ $x_1)_{Theme})$

In other words:

[a] Birds are always vertebrates.
[b] A typical bird has feathers, two legs and two wings.
[c] A typical bird flies.

Our analysis starts with the syntactico-semantic description of participants, i.e. the smallest conceptual representation structures within predications. Just like in Dik's Functional Grammar, there are two types of participants: arguments and satellites. In both cases, their syntactic representation is as follows:

$$\Lambda \equiv \left(\sum_{i=0}^{1} \alpha \;\; \Pi : \; \sum_{j=0}^{n} \Xi \right)_{\omega}$$

$\alpha$ = [1 | 2 | 3 | 4 ... m | s | p | i]
$\Pi$ = [x | f]
$\omega$ = [Agent | Theme | Referent | Location | Origin | Goal | Beneficiary | Attribute | Means | Company | Instrument | Role | Route | Comparison | Manner | Scene | Reason | Distance | Time | Time_from | Time_to | Duration | Frequency | Purpose | Result | Coocurrence | Sequence | Condition | Speed | Location_in | Location_out]

That is, $\Lambda$ is a participant whose type is specified by $\Pi$, where indexed labels *x* and *f* are used by arguments and satellites respectively. A participant can be preceded by an operator ($\alpha$), which applies a specific kind of quantification to the concept expressed as a selection preference (table 1):

| Feature | Value |
|---|---|
| absolute quantifier | 1 \| 2 \| 3 \| 4 ... |
| relative quantifier | m \| s \| p |
| indefinite quantifier | i |

**Table 1. Participant operators.**

If a quantification operator is incorporated, this will be as restricted as possible. In other words, if we refer to one entity, then operator *1* will always be used. Some alternatives arise when the selection preference is associated to more than one referent. In these cases, our decision is based on the parameters established by table 1. That is, if the exact number of entities is known, then an absolute quantifier is used; otherwise, we try to state if there are many (*m*), some (*s*) or just a few (*p*) referents. Finally, if none of these criteria can be applied, then operator *i* is chosen by default, which is read as 'it refers to more than one entity, but we don't know how many'.

These quantification operators are not obligatory, because there are some cases where it is impossible to apply one particular operator, e.g. when the concept to be quantified refers to an uncountable entity (e.g. air), or the selection preference is an attribute or a predication.

When selection preferences (Ξ) are stated, a colon (:) is used to separate them from their type of participant (Π), that is, argument or satellite. Selection preferences in a participant can be expressed by means of a predication, or one or more basic concepts from the entity or attribute subhierarchies in the ontology. Multiple selection preferences within a participant are linked by no logical connector, because they are handled as items of a list. In this case, a comma is used as a separator for selection preferences.

Moreover, every participant performs one and only one semantic function (ω). Regarding the inventory of semantic functions in FungramKB, some are argument-oriented (e.g. *Agent, Theme, Referent*), others are satellite-oriented (e.g. *Means, Company, Instrument, Role, Route, Comparison, Manner, Scene, Reason, Distance, Time, Time_from, Time_to, Duration, Frequency, Location_in, Location_out, Purpose, Result, Coocurrence, Sequence, Condition, Speed*), and finally there are some functions which can be used indistinctively for both arguments and satellites (e.g. *Location, Origin, Goal, Beneficiary, Attribute*).

Dik's Functional Grammar proposes using lexical units from the own language when describing meaning postulates, since meaning definition is an internal issue of the language. However, this strategy contributes to lexical ambiguity in representation due to the polysemic nature of the defining lexical units. In addition, describing the meaning of lexical units in terms of other lexical units leads to some linguistic dependency. Instead, FunGramKB uses concepts for the formal description of meaning postulates.

On the other hand, the syntactic representation of a predication is as follows:

$$\Delta \equiv \delta\ (e{:} \sum_{k=0}^{5} \beta\ \Gamma\ \sum_{l=0}^{n} \Lambda)$$

$\delta = [+ \mid *]$

$\beta = [ing \mid pro \mid egr \mid rpast \mid npast \mid pres \mid nfut \mid rfut \mid cert \mid prob \mid pos \mid obl \mid adv \mid perm \mid n]$

That is, a predication Δ is stated by an indexed *e*, which is always followed by a colon and an event (Γ) from the basic coneptual level of the ontology. Between the colon and the event concept, it is possible to insert one or more operators, up to five. Each one of these operators conveys a different feature (table 2)[1]:

| Feature | Value |
|---|---|
| Aspectuality | ing \| pro \| egr |
| Temporality | rpast \| npast \| pres \| nfut \| rfut |
| Epistemic modality | cert \| prob \| pos |
| Non-epistemic modality | obl \| adv \| perm |
| Polarity | n |

**Table 2. Predication operators.**

Aspectuality operators show the distinctions occuring in the development of an event, in terms of the start-continuation-end of the event: e.g. ingressive (*ing*), progressive (*pro*) and egressive (*egr*). In other words, these operators show a temporal component which is relevant in the internal development of the state of affairs, instead of positioning the event in a time axis.

Temporality operators are used to position the state of affairs designated by the predication in some interval along the time axis: e.g. remote past (*rpast*), near past (*npast*), present (*pres*), near future (*nfut*) and remote future (*rfut*).

With regard to modality, two broad categories of operators are distinguished, i.e. epistemic and non-epistemic, depending on if the speaker evaluates or not the truth of the predication. In FunGramKB, epistemic modality operators indicate the certainty (*cert*), probability (*prob*) or possibility (*pos*) that the information represented in the predication is true from the speaker's point of view. On the other hand, non-epistemic modality operators reflect a strong deontic nature, stating the obligation (*obl*), advise (*adv*) or permission (*perm*) of the information in the predication.

Our polarity operator *n* is similar to predicate *neg* in d-Prolog [6], since negative information can be explicitly stated. Therefore, it is different from the built-in operator *not* in Prolog, which indicates negation by failure.

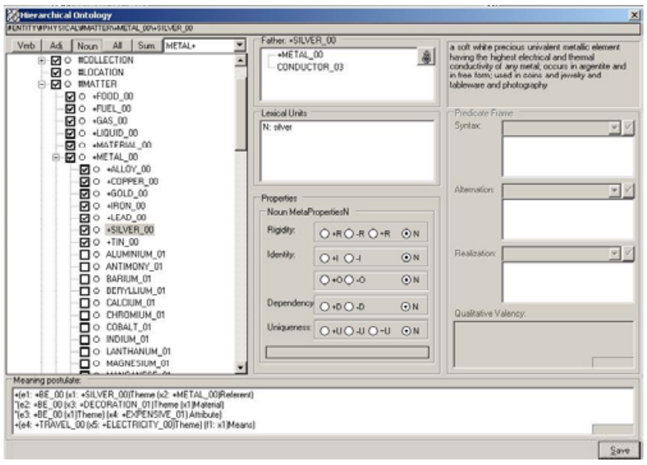
Finally, each predication taking part in a meaning postulate is preceded by a reasoning operator (δ) in order to state if the predication is strict (+) or defeasible (*). Our inference engine handles predications as rules, allowing monotonic reasoning with strict predications, and non-monotonic with defeasible predications. Strict predications are law-like rules, which have no exceptions: e.g. whales are mammals, circles are round. On the other hand, defeasible predications, which make up our commonsense, can be withdrawn in the light of some more specific predication, which can be strict or defeasible. Thus, defeasible predications are rules which perform inference allowing contradictory information to override them: e.g.

[1] This table is not intended to provide an exhaustive list of predication operators.

birds typically fly. Defeasible reasoning allows the possibility of working with incomplete information, so a closed-world assumption cannot be applied.

There are three logical connectors used in FunGramKB: conjunction (&), disjunction (|) and exclusion (^). Conjunction is the default logical connection, so it is also effective when two participants or predications are simply concatenated with no explicit connector.

The construction of meaning postulates in FunGramKB is semiautomatic, because human intervention is required, but the lexicographer's intuition is guided and reviewed through our lexicographical tool, so that consistent well-formed predications can be stored (figure 2).



#ENTITY#PHYSICAL#MATTER+METAL_00+SILVER_00

a soft white precious univalent metallic element having the highest electrical and thermal conductivity of any metal; occurs in argentite and in free form; used in coins and jewelry and tableware and photography

N: silver

+(e1: +BE_00 (x1: +SILVER_00)Theme (x2: +METAL_00)Referent)
*(e2: +BE_00 (x3: +DECORATION_01)Theme (x1)Material)
*(e3: +BE_00 (x1)Theme) (x4: +EXPENSIVE_01) Attribute)
+(e4: +TRAVEL_00 (x5: +ELECTRICITY_00)Theme) (f1: x1)Means)

**Figure 2. Ontology editor in FunGramKB.**

When predications are built for a NLP knowledge base in order to represent an average adult speaker's linguistic competence, dictionaries must be our guide [5]. Dictionaries are reliable repositories of information that several generations of expert speakers have judged to be relevant for lexical meaning. In that respect, FunGramKB works with several machine-readable dictionaries, *Collins COBUILD English Dictionary* [7], *Oxford Advanced Learner's Dictionary* [2] and *Longman Web Dictionary* [4], and the linguistic taxonomy *WordNet 1.6*. Ide and Véronis [3] recommend using several dictionaries as sources of lexical acquisition, because what a particular dictionary lacks is usually supplied by another dictionary.

## 3.3: Computational implementation

In this section, we describe the computational implementation of meaning postulates in FunGramKB. The first stage of this process was the formal definition of the theoretical model explained in section 3.2 by means of a context-free grammar, resulting in table 3[2].

| | | |
|---|---|---|
| <S> | ::= | <S><CON><Se>\|<Se> |
| <Se> | ::= | <Se><STRe>\|<STRe> |
| <STRe> | ::= | (<S>)\|<OPr>(e<N>:<OPe><LBCv> <Lx><Lf>)\|<OPr>(e<N>:<OPe> <LBCv><Lx>) |
| <LBCv> | ::= | *<List of basic concepts_event>* |
| <OPe> | ::= | <Lasp>\|<Ltem>\|<Lmode>\| <Lmodne>\|<OPneg> |
| <Lasp> | ::= | <OPasp><Ltem>\|<OPasp><Lmode> \|<OPasp><Lmodne>\|<OPasp> <OPneg>\|<OPasp> |
| <Ltem> | ::= | <OPtem><Lmode>\|<OPtem> <Lmodne>\|<OPtem><OPneg>\| <OPtem> |
| <Lmode> | ::= | <OPmode><Lmodne>\|<OPmode> <OPneg>\|<OPmode> |
| <Lmodne> | ::= | <OPmodne><OPneg>\|<OPmodne> |
| <OPasp> | ::= | ing\|pro\|egr |
| <OPtem> | ::= | rpast\|npast\|pres\|nfut\|rfut |
| <OPmode> | ::= | cert\|prob\|pos |
| <OPmodne> | ::= | obl\|adv\|perm |
| <OPneg> | ::= | n |
| <Lx> | ::= | <Lx><CON><X>\|<X> |
| <X> | ::= | (<OPxf> x<N><LSPx>)<SFx>\| (<OPxf> x<N>)<SFx>\|(<OPxf>) <SFx> |
| <LSPx> | ::= | :<LBCea>\|:<S> |
| <SFx> | ::= | Agent\|Theme\|Referent\|Goal\| Beneficiary\|Attribute\|Location\| Origin |
| <LBCea> | ::= | *<List of basic concepts_entity_ attribute>* |
| <Lf> | ::= | <Lf><CON><Sf>\|<Sf> |
| <Sf> | ::= | <Sf><FF>\|<FF> |
| <STRf> | ::= | (<Lf>)\|(<OPxf> f<N>:<LSPf>) <SFf> |
| <LSPf> | ::= | <LBCea>\|<S>\|x<N> |
| <SFf> | ::= | Beneficiary\|Company\|Instrument\| Role\|Origin\|Route\|Goal\|Comparison\| Manner\|Location\|Scene\|Reason\| Distance\|Time\|Time_from\|Time_to\| Duration\|Frequency\|Purpose\|Result\| Coocurrence\|Sequence\|Condition\| Attribute\|Means\|Speed\|Location_in\| Location_out |
| <N> | ::= | <N><D>\|0\|1\|2\|3\|4\|5\|6\|7\|8\|9 |
| <D> | ::= | 0\|1\|2\|3\|4\|5\|6\|7\|8\|9 |
| <CON> | ::= | \|\|^\|&\|λ |
| <OPr> | ::= | +\|*\|λ |
| <OPxf> | ::= | <N>\|m\|s\|p\|i |

**Table 3. Context-free grammar for meaning postulates.**

The second stage involved the election of a language for knowledge representation in FunGramKB. XML was chosen because it helps the system with the establishment of premises for structured data transfer, the autonomous separation of knowledge from its representation formalsism, and the atomization of the various components the meaning postulate is made up of, speeding up the

[2] Strictly speaking, the grammar in this table is not well-formed, because symbol λ is not used within the start-symbol axiom. However, this decision was taken in the interests of a more expressive clarity.

inference process and the direct access to particular conceptual units in the meaning postulate. One of the first steps in this second stage was the conversion of the context-free grammar in table 3 into an XML schema, so that the application could recognize well-formed meaning postulates. To illustrate, figure 3 displays the XML representation of some predications in the meaning postulate of concept +SILVER_00:

**SILVER_00**
+($e_1$: +BE_00 ($x_1$: +SILVER_00)$_{\text{Theme}}$ ($x_2$: +METAL_00)$_{\text{Referent}}$)
*($e_2$: +BE_00 ($x_1$)$_{\text{Theme}}$ ($x_3$: +EXPENSIVE_00)$_{\text{Attribute}}$)
+($e_5$: +TRAVEL_00 ($x_4$: +ELECTRICITY_00)$_{\text{Theme}}$ ($f_1$: $x_1$)$_{\text{Means}}$)

```
<S>
    <e N="1" OPr="+">
        <LBCv>+BE_00</LBCv>
        <x N="1" SFx="Theme">
            <LBCea>+SILVER_00</ LBCea >
        </x>
        <x N="2" SFx="Referent">
            <LBCea>+METAL_00</LBCea>
        </x>
    </e>
    <e N="2" OPr="*">
        <LBCv>+BE_00</LBCv>
        <x N="1" SFx="Theme"/>
        <x N="3" SFx="Attribute">
            <LBCea>+EXPENSIVE_00</LBCea>
        </x>
    </e>
    <e N="3" OPr="+">
        <LBCv>+TRAVEL_00</LBCv>
        <x N="4" SFx="Theme">
            <LBCea>+ELECTRICITY_00</LBCea>
        </x>
        <f N="1" SFf="Means">
            <x>1</x>
        </f>
    </e>
</S>
```

**Figure 3. Meaning postulate for +SILVER_00 in XML.**

## Acknowledgements

FunGramKB is part of the research carried out in the project 'Construcción de un lexicón multipropósito para el procesamiento del lenguaje natural' funded by Universidad Católica San Antonio de Murcia, code number PMAFI-PI-03/1C/03.

## References

[1] Dik, S.C. 1997 (1989). *The Theory of Functional Grammar*. Berlin-New York: Mouton de Gruyter.

[2] Hornby, A.S. 2003. *Oxford Advanced Learner's Dictionary of Current English*. Oxford: Oxford University Press. [http://www.oup.com/elt/global/products/oald]

[3] Ide, N. and J. Véronis. 1994. "Machine readable dictionaries: what have we learned, where do we go?". *Proceedings of the Post-Coling 94 International Workshop on Directions of Lexical Research*. 137-146.

[4] Longman Group. 2001. *Longman Web Dictionary*. Harlow: Longman. [http://www.longmanwebdict.com]

[5] Marconi, D. 1997. *Lexical Competence*. Cambridge-Massachusetts: the MIT Press.

[6] Nute, D. 1993. "Defeasible Prolog". *Working Papers of the 1993 AAAI Fall Symposium on Automated Deduction and Nonstandard Logics*. Menlo Park: AAAI Press. 105-112.

[7] Sinclair, J., ed. 1995 (1987). *Collins COBUILD English Dictionary*. London: Collins.

[8] Velardi, P., M.T. Pazienza and M. Fasolo. 1991. "How to encode semantic knowledge: a method for meaning representation and computer-aided acquisition". *Computational Linguistics* 17, 2. 153-170.