Computational NSM: a PROLOG-based notation

Francesco Zamblera

August 13, 2010

Abstract

In this paper, a notation is developed for representing NSM texts using PROLOG clauses. It is not so much an attempt to "formalize" NSM, as the first step towards the development of a set of computational tools which could be useful for research in the NSM paradigm.

In the attempt to translate NSM into a language-independent format, some proposals are made about the structure of "substantive phrases" and a basic "tense-aspectual system". These could be of interest for NSM research in any case.

Introduction

The Natural Semantic Metalanguage is an important tool for semantic research which has proved itself very useful in many such different domains as lexicography[1], ethnopragmatics[2], anthropology[3], Biblical exegesis[4].

In this article, a notation is developed for writing "NSM formulas", that is, a representation of NSM texts independent of their particular NSM-language-particular incarnation. These formulas are written in a PROLOG-derived notation, which I will call NSM-PROLOG notation. (No knowledge of PROLOG or computational linguistics is assumed on the part of the reader).

The NSM-PROLOG notation is being developed together with some PROLOG tools:

- `nsm-gen-pl` was my first attempt to a NLP tool for NSM. It is a program which, after having loaded a PROLOG grammar of a language-specific NSM, can read NSM-PROLOG formulas and generate the equivalent sentences in that language. The format of the English grammar is briefly described in Zamblera (2010a). The program (which is still available from the author's website) has been superseded by `nsm-dalia-pl`, which can both generate natural language sentences from NSM formulas and parse sentences into formulas.

- `nsm-dalia-pl` is both a parser and a generator. Is uses a fairly standard PROLOG parsing technique (namely, definite clause grammars), and

[1] Wierzbicka (1987).
[2] Goddard (2006).
[3] Goddard & Wierzbicka (2004), Wierzbicka (2002a).
[4] Wierzbicka (2001).

another program for generating allolexes, `hist-morph.pl`. The allolex-generating tool can be used also for describing diachronic changes in groups of closely related languages with a unique database, as shown in Zamblera (2010b)[5].

The NSM-PROLOG notation has been developed together with the first program, the generator. After having selected (somewhat arbitrarily) as the corpus for this work the English NSM texts contained in Wierzbicka (2001), I have tried to account for all the morphosyntactic phaenomena presented by those texts. The program has been tested successfully against all the NSM texts contained in the first part of Wierzbicka (2001, pp. 21-225), which were translated by myself into the NSM formula notation, and rewritten back into English NSM by the computer. Examples are provided in this article (see section 8).

Developing computational tools for the NSM can be a fruitful exercise both for NSM research and computational linguistics:

- On the NSM-theoretic side, all assumptions have to be made explicit as a computational model of the theory is being developed. For example, tools must be provided for the selection of the right allolex in the proper context, inflection and agreement phaenomena must be worked out fully. In this paper, some proposal for the structure of substantive phrases (see section 3) and the tense-aspectual system (see section 5.2) emerge quite naturally in order to describe adequately the English NSM texts in the corpus;

- on the computational side, researchers in natural language parsing and generation know very well the complexities of building a real-world application which has to function with real data. NSM is a mini-language which, on the one hand, is (lexically and syntactically) restricted enough to avoid many of the complications found in processing unrestricted specimina of natural language; on the other hand, NSM represents a complete subset of a natural language, which renders the developement of a computational system a non-trivial matter. As the grammar of the NSM subset of a particular language can be thought of as the core gramamr of that language (see e.g. Wierzbicka, 2002b), any computational tool which is miminally adequate should be developed, first of all, to work with such a subset.

1 The Natural Semantic Metalanguage

Goddard (2008a, p. 1) defines NSM as

> a decompositional system of meaning representation based on empirically established universal semantic primes, i.e. simple indefinable meanings which appear to be present as identificable word-meanings in all languages.

[5] I am developing another NLP tool for automatic translation between two NSMs, in javascript, called *nsm-dalia-js*. This program does not use the NSM formulas described in this paper. As for *nsm-dalia-pl*, a previous version, written with dependency-grammar based modules for English and Tok Pisin is also available on the author's website. The definite-clause-grammar version, however, is by far simpler.

A tool for representing meaning, NSM is based on natural languages, and not on a constructed formal system. Quoting further from Goddard (2008b, p. 3), this is based on

> the conviction that ordinary natural languages are adequate to represent their own semantics via language-internal paraphrase,

and on the fact, repeatedly stated in literature about NSM, that any formal language specially devised for semantic analysis must rely on natural language in order to be understood by the researchers themselves.

Therefore,

> an optimal semantic metalanguage must be based as transparently as possible on ordinary natural languages, and it must consist only of elements whose meanings are present in all natural languages, i.e. of universally lexicalised meanings. In short, it ought to be based on "the intersection of all languages[6]."

To this intersection belong:

- about sixty conceptual primes, such as `SOMEONE, SOMETHING, I, YOU, GOOD, SEE, THINK, LIVE`;
- the combinatorial properties of these primes; for example, such syntactic frames as `SOMETHING GOOD`, `SOMEONE LIVES`. *Canonical sentences* are those combinations of prime which include minimally the combination of a "substantive phrase" with a "verb phrase" in a subject-predicate construction.[7]

At present, the list of recognised primes includes some sixty items, shown in table 1, taken from Goddard (2008b, p. 33)[8].

These primes are supposed to have an exact equivalent in every language. However, language-particular exponents of the primes can be affected by *polysemy* and/or *allolexy*.

Polysemy: A language-particular exponent of a prime is *polysemous* when it has also other meanings. For example, the English verb *think* is the exponent of the prime `THINK`, and has other meanings as well, such as the one in the "opinion frame" (*She thinks that* __).[9]

The English verb *think* is the exponent of `THINK` when it is used in one of its *canonical frames*[10]:

1. `SOMEONE THINKS ABOUT SOMEONE/SOMETHING`

[6] Goddard (2008b), p. 5

[7] "We can think of a simple NSM clause as consisting of a predicate, such as HAPPEN, DO, SAY, THINK, or WANT, together with one or more substantive phrases whose nature is constrained by the identity of the predicate." (Goddard, 2008b, p. 13).

[8] Cliff Goddard (p.c., July 2010) informed me that current research in NSM adds two new primes: FEW (a little) and clausal AS (this thing happened, *as I wanted it*)

[9] As Goddard (p.c., November 2008) points out, "we now believe that this frame is only possible when it is tied to a time adjunct. It can't be used (in NSM syntax) in "opinion-giving" contexts such as 'Some people think that Elvis is still alive', even though such sentences are perfectly OK in ordinary English." Goddard & Karlsson (See 2008).

[10] Goddard (2008b), 14.

Table 1: Semantic primes

Substantives : I, YOU, SOMEONE, SOMETHING/THING, PEOPLE, BODY

Relational substantives : KIND, PART

Determiners : THIS, THE SAME, OTHER/ELSE

Quantifiers : ONE, TWO, MUCH/MANY, SOME, ALL

Evaluators : GOOD, BAD

Descriptors : BIG, SMALL

Mental predicates : THINK, KNOW, WANT, FEEL, SEE, HEAR

Speech : SAY, WORDS, TRUE

Actions, events, movement, contact : DO, HAPPEN, MOVE, TOUCH

Location, existence, possession, specification : BE (SOMEWHERE), THERE IS, HAVE, BE (SOMEONE/SOMETHING)

Life and death : LIVE, DIE

Time : WHEN/TIME, NOW, BEFORE, AFTER, A LONG TIME, A SHORT TIME, FOR SOME TIME, MOMENT

Space : WHERE/PLACE, HERE, ABOVE, BELOW, FAR, NEAR, SIDE, INSIDE

Logical concepts : NOT, MAYBE, CAN, BECAUSE, IF

Intensifier, augmentor : VERY, MORE

Similarity : LIKE

2. `SOMEONE THINKS SOMETHING GOOD/BAD ABOUT SOMEONE`
3. `SOMEONE THINKS LIKE THIS: ‘‘ __ ’’`
4. `SOMEONE THINKS THAT +` *‘‘tensed’’ sentence*, that is, sentence with a time adjunct.

Allolexy: Language-particular exponents of primes are affected by such phenomena as gender-number agreement, inflection for tense and aspect, or selectional restriction. For example, the prime `SOMETHING` has in English the allolex *things* in the frame `TWO GOOD __` .

Such allolexy is a pervasive fact of human languages. As Goddard (2008a, p. 7) observes,

> As far as we know, there is no human language in which exponents of semantic primes are unaffected by polysemy and allolexy,

i.e. in no human language there is a transparent one-to-one mapping between universal primitive meanings and surface lexical forms.

NSM sentences can combine to form texts. Prime combinations such as `AT THE SAME TIME`, `BECAUSE OF THIS`, as well as the use of indentation, ensure textual cohesion.

2 The NSM-PROLOG notation

Of course, if an NSM sentence can be "expressed" in PROLOG, then NSM is "formalizable". This only means, of course, that an NSM is translatable into a man-made formal language, which still needs natural language to be interpreted by people: for an human reader to understand something like

`sp(this,e,e,good,something)`

means to mentally translate it into "this thing" (or "esta cosa", or any other equivalent).

The NSM-PROLOG notation which now will be developed is not meant to be an underlying model of NSM nor a model of what is effectively stored in speaker's minds, but only a "translation" of the "Natural Semantic Metalanguage into PROLOG", a computer language, in order for the computer to be able to "understand" NSM; that is, in order for us to be able to develop computational tools for NSM.

2.1 Predicates

Phrases and sentences will be represented by a uniform notation, which corresponds to the PROLOG representation of a predicate. A PROLOG predicate consists of a *functor* which can (but need not) take *arguments*. If it has, it is followed by its arguments enclosed in parentheses (and separated by commas if there is more than one of them). If it has not, it is called an *atom*. The functor can have an arbitrary name, but it *must* start with a lower-case letter, because a word starting with an upper-case letters represents a *variable* in PROLOG. The whole must be followed by a fullstop.

In s PROLOG predicate, the number and place of argument slots is distinctive:

```
pred(arg1,arg2).
```

and

```
pred(arg1,arg2,arg3).
```

count as different predicates, although their functors have the same name, whereas the two structures

```
pred(arg1,arg2).
```

and

```
pred(arg2,arg1).
```

have the same predicate, but the role of the arguments is inverted. For example, suppose we want to represent an English transitive verb such as *see* with a two-place PROLOG predicate:

```
see(Seer,Seen).
```

Now, the structure `see(john,mary)` will represent the clause *John sees Mary*, while `see(mary,john)` will stand for *Mary sees John*.

If we decided to represent this (simplified) semantic structure of "see" with:

```
see(Seen,Seer).
```

then the structure `see(john,mary)` would represent "Mary sees John". In other words, the programmer decides which semantics he has to attribute to the predicate slots – of course, we must be consistent: if we deside that "see" is `see(Seer,Seen)`, we had better represent "eat" as `eat(Eater,Eaten)` rather that `eat(Eaten,Eater)`.

The predicate structure is *recursive*, that is, arguments of a predicate can be predicates with arguments, and so on. Some examples:

```
pred.
pred(arg).
pred(arg1,arg2,arg3).
pred1(arg1,arg2,pred2(arg3,arg4)).
```

Example of NSM primes represented by atoms[11]:

- `something.`
- `someone.`
- `good.`
- `feel.`

An example of a NSM-PROLOG predicate-argument structure is the *substantive phrase*, represented by the predicate `sp` along with its five arguments in the following order:

1. a determiner such as `this`;
2. one of the two primes `same` or `other`;
3. a quantifier, such as `one`, `two`, `many`;
4. an attribute, such as `good`, `bad`, `big`, `small`;
5. the head-noun.

An empty slot is filled by the special predicate `e`. So, for example, the substantive phrase `TWO GOOD THINGS` will be represented by the following formula:

[11] In NSM literature, primes are often designed in small caps. But as in PROLOG any word beginning with a capital letter is considered a *variable* and not an atom, NSM-PROLOG notation uses lower-case letters instead.

```
sp(e,two,e,good,something).
```

where the determiner slot is empty, as well as the same-or-other slot. I will often use this format in presenting the NSM-PROLOG predicate frames, with variables as slot names:

$$sp(Determiner, SameOrOther, Quantifier, Evaluator, Head).$$

2.2 Lists

The other PROLOG structure used in the NSM-PROLOG notation is the *List*. A list is a sequence of comma-separated elements enclosed in brackets. Elements of a list can be atoms, predicate-argument structures (however complex), or other lists. The *empty list*, a list without any elements, is represented by `[]`. Examples:

- `[].`
- `[a,b,c].`
- `[a,b,pred1(arg1,arg2,pred2(arg3)),p,q].`
- `[a(b,c),[d,e,f],g,[]].`

.

Lists are used in NSM-PROLOG for two purposes:

- to represent the *valence* of NSM "verbal" predicates (section 4),
- to represent *sentence groups* (section 7.1).

In the following discussion of the PROLOG-NSM notation, I will borrow from tagmemics both the useful device of analysing the syntactic structure of NSM at different levels (phrase, clause, sentence and sentence group) and the concepts of of *slot* and *filler*. The building blocks of NSM-PROLOG notation are PROLOG predicates containing slots that can be filled by other predicates. Each slot corresponds to a specific *function* in the predicate frame (see e.g. the discussion of the `sp` preedicate above). And each predicate belongs to a specific level:

LEVEL	PREDICATES
phrase	`sp`, `p`
clause	`s`
sentence	`s`, `if`, `when`, `because`
sentence group	`sg`

In the following section, each level will be examined in turn.

3 Phrase Level (1): Substantive Phrase

As we have seen, in NSM-Prolog notation, a substantive phrase is represented as

(1) $sp(Determiner, Quantifier, SameOrOther, Attribute, Head).$

3.1 Head

The *Head* slot can be filled by

1. one of the substantives `someone (person), people, something (thing), somewhere, space, time`;
2. one of the relational substantives `kind`, `part`, `body` with their complement (a full substantive phrase for `body`, a bare substantive for `kind` and `part`):

   (2) $kind(Complement); part(Complement); body(Complement)$

Examples:

NSM-Prolog	English NSM
`someone`	*someone, person*
`something`	*something, thing*
`kind(something)`	*a kind of thing*
`part(something)`	*a part of something*
`body(sp(this,e,e,e,someone))`	*this person's body*

3.2 Attribute

The *Attribute* slot can be filled by:

1. One of the NSM attributes: evaluators (`good, bad`, descriptors (`big, small`), and perhaps the space-related primes FAR and NEAR (I don't know whether their attributive use is part of NSM; I mean combinations like A FAR PLACE, A NEAR PLACE);
2. A structure of the form:

   (3) $very(Attribute)$

   with the slot *Attribute* filled by one the same NSM attributes.
3. The one-place predicate `like` with its complement:

   (4) $like(Complement)$

   where the slot *Complement* is filled by one of the substantives `me, you`, or, recursively, by another substantive phrase.

Examples:

NSM-Prolog	English NSM
`good`	*good*
`small`	*small*
`very(big)`	*very big*
`like(me)`	*like me*
`like(sp(this,e,e,e,someone))`	*like this person*

3.3 Same, Other, Quantifiers

The *SameOrOther* slot can be filled by one of the two primes `same` or `other`, while *Quantifier* can be filled by one of the NSM quantifiers `one, two, some, many, all` (see examples in section 3.5).

I have allocated a separate slot for SAME and OTHER because (OTHER at least) can cooccur with other determiners, as in "this other person".

3.4 Determiner

Current NSM research recognises `this, the same` and `other` as determiners. I tentatively add to this list the atom `any`, in order to account for the difference between such English-NSM expressions as *the other thing* vs *another thing*, or Russian-NSM *kto-to* vs *kto-nibud'* (both someone, but in a specific vs. generic reading)[12]. Of course, it remains to be seen whether such a distinction a) is useful for NSM explication and b) is universal.

3.5 Examples of substantive phrases

Each slot which can be filled by a substantive phrase can also be filled by one of the two primes I and YOU, represented by the two predicates `me` and `you`. Accordingly, a substantive phrase in NSM-PROLOG consists of one of the following predicates:

- the atoms `me` and `you`;
- the predicate `sp` with its five arguments.

NSM-PROLOG	ENGLISH NSM
`sp(e,e,e,e,something)`	*something*
`sp(e,e,e,e,someone)`	*someone*
`sp(e,e,e,good,something)`	*something good*
`sp(e,e,e,big,someone)`	*someone big*
`sp(this,e,e,bad,something)`	*this bad thing*
`sp(e,many,e,e,something)`	*many things*
`sp(this,two,other,e,something)`	*these two other things*
`sp(this,e,e,e,somewhere)`	*this place*
`sp(e,e,e,like(you),someone)`	*someone like you*
`sp(any,e,other,e,someone)`	*another person*
`sp(e,e,other,e,someone)`	*the other person*
`me`	*I, me, my*
`you`	*you, your*

4 Phrase Level (2): Predicate Phrase

The notion of *valence* is captured in NSM-PROLOG notation by the two-place predicate *p*, which takes as argument a predicate and a PROLOG list :

[12]This predicate is probabily superfluous, as the generic vs. specific reading could be read off the syntactic frame in which the substantive phrase appear, cfr. the distinction, introduced in Goddard (2008b, 13), between *complements* and *arguments*.

(5) $p(Predicate, ArgumentList).$

The *ArgumentList* slot is filled by a PROLOG list containing the arguments of the prime which fills the *Predicate* slot. All predicates restrict the number of the items in the argument list. In the following subsections, predicates will be classified as to the number of arguments they select.

4.1 One-argument predicates

Active predicates: MOVE, DIE. The NSM-PROLOG representation of the primes MOVE and DIE is:

(6) $p(Predicate, [Event]).$

where the *Predicate* slot is filled by either `move` or `die`.
Examples:

NSM-PROLOG	ENGLISH NSM
`p(move, [sp(e,e,e,e,something)])`	*(for) something to move*
`p(move,[me])`	*(for) me to move*
`p(die,[sp(e,e,e,e,someone)])`	*(for) someone to die*

Descriptors: BIG, SMALL. They have the same structure as the preceding predicates:

(7) $p(Descriptor, [Argument])$

where the slot *Descriptor* is filled by `big` or `small`, or by `very(big)`, `very(small)` . Examples:

NSM-Prolog	English NSM
`p(small, [sp(this,e,e,e, something)])`	*(for) this thing (to be) small*
`p(big, [me])`	*(for) me (to be) big*
`p(very(small),` `[sp(this,e,e,e, somewhere)])`	*(for) this place (to be) very small*

4.2 Two-argument predicates

These have the general structure:

(8) $p(Predicate, [Arg1, Arg2]).$

4.2.1 HAPPEN

The second, facultative, argument of HAPPEN can be a benefactive (or "malefactive"), like in "something happened to this person", or a locative, as in "something good happened in this place".

(9) $p(happen, [Arg1, Arg2]).$

Examples:

NSM-Prolog	English NSM
`p(happen, [sp(this,e,e,e,something), sp(this,e,other,e,someone) ])`	*(for) this thing to happen to this other person*
`p(happen, [sp(e,e,e,e,something), me])`	*(for) something to happen to me*
`p(happen, [sp(e,e,e,good,something), e])`	*(for) something good to happen*

4.2.2 FEEL

The prime FEEL has a SOMETHING-complement (FEEL SOMETHING (GOOD/BAD)), and can have an optional target argument (FEEL SOMETHING TOWARDS SOMEONE/SOMETHING).

Examples:

NSM-Prolog	English NSM
`p(feel, [sp(this,e,e,e,someone), sp(this,e,e,good,something) ])`	*(for) this person to feel something good*
`p(feel, [sp(e,e,e,e,someone), sp(e,e,e,e,something), me])`	*(for) someone to feel something towards me*

4.2.3 (BE) LIKE

(10) $p(like, [Arg1, Arg2]).$

Examples:

NSM-Prolog	English NSM
`p(like, [sp(this,e,e,e,something), sp(this,e,other,e,something) ])`	*(for) this thing (to be) like this other thing*
`p(like, [you, me])`	*(for) you (to be) like me*

4.2.4 Locational predicates

NSM locational predicates FAR, NEAR, ABOVE, BELOW, INSIDE are represented by the atoms `far, near, above, below, inside`. NSM-PROLOG

provides two notations for predicate phrases with locationals. The first is simply:

(11) $p(LocationalPred, [Arg1, Arg2])$

Examples:

NSM-PROLOG	ENGLISH NSM

`p(far, [sp(this,e,e,e,somewhere),`
`  sp(this,other,e,e,somewhere)`
`])`
	(for) this place (to be) far from this other place

`p(near,[you,me])`	*(for) you (to be) near me*
`p(far, [sp(this,e,e,e,somewhere), e])`	*(for) this place (to be) far*

`p(inside, [sp(e,e,e,e,something),`
`  sp(e,other,e,e,something)`
`])`
	(for) something (to be) inside of something else

`p(very(far), [sp(this,e,e,e,somewhere), e])`	*(for) this place (to be) very far*

The second possibility is to analyze a locational predicate as an instance of BE (SOMEWHERE) with a locational complement, as proposed by Goddard (2002, 308):

(12) $p(be, [Locatum, locationalPredicate(Locus)])$

Example:

NSM-Prolog	English NSM
`p(be,[sp(this,e,e,e,someone), above(me)])`	*(for) this person (to be) above me*
`p(be,[sp(this,e,e,e,somewhere),` `  above(sp(this,e,other,e,somewhere))])`	*(for) this place (to be) above this other place*

More examples in section 4.3.5.

4.2.5 Evaluators: GOOD and BAD.

Evaluators have a *Benefactive* slot (GOOD FOR..., BAD FOR...), which can be filled by the predicates `me`, `you` and `sp`[13]:

[13] And `e`, when we want to represent a phrase without any benefactive argument.

(13) $p(Evaluator, [Theme, Benefactive])$

The *Evaluator* slot can be filled by the atoms `good`, `bad`, or by the predicate structures `very(good)`, `very(bad)`.

Examples:

NSM-Prolog	English NSM
`p(good,[sp(this,e,e,e,something),e]))`	*(for) this thing to be good*
`p(good,[sp(this,e,e,e,something),me]))`	*(for) this thing to be good for me*
`p(very(bad),[sp(e,e,e,e,something)]))`	*(for) something to be very bad*

4.2.6 Other transitive predicates

TOUCH, SEE, HEAR, and WANT have the same structure (for WANT, however, there are three options, see below):

(14) $p(Pred, [Arg1, Arg2])$

Examples:

NSM-Prolog	English NSM
`p(touch,[me, sp(e,e,e,e, something)])`	*(for) me to touch something*
`p(see,[you, sp(this,e,e, e,something)]).`	*(for) you to see something*
`p(hear,[sp(this,e,e, e,someone),word(you)]).`	*(for) this person to hear your words*

WANT. The *Arg2* slot of the predicate `want`, can be filled either by a substantive phrase, or by another *p* structure. In the latter case, if the first argument of the *p* structure is `e`, the formula represents an *equi* structure. So we have three possibilities for `want`:

- Transitive structure (X WANTS SOMETHING):

  (15) $p(want, [SP_{subj}, SP_{obj}])$

  Examples:

NSM-Prolog	English NSM
`p(want,[me, sp(this,e,e,e,something)])`	*(for) me to want this thing*
`p(want,[sp(this,e,e,e,someone), sp(e,e,e,e,something)])`	*(for) this person to want something*

- Equi structure (X WANTS TO DO SOMETHING):

  (16) $p(want, [Subject, p(Pred, [e, ...])])$

Examples:

NSM-Prolog	English NSM
`p(want, [me, p(see,[e, sp(this,e,e,e,something)])])`	*(for) me to want to see this thing*
`p(want, [sp(this,e,e,e,someone), p(know,[ e, sp(e,e,e,e,something])])]`	*(for) this person to want to know something*

- Non-equi structure (X WANTS Y TO DO SOMETHING):

(17) $p(want, p(Subject1, [p(Pred, [Subject2, ...])])$

where *Subject1* and *Subject2* are different.

Examples:

NSM-Prolog	English NSM
`p(want,[me, p(see, [you, sp(this,e,e,e,something) ]) ])`	*(for) me to want you to see this thing*
`p(want, [sp(this,e,e,e,someone), p(know, [me, sp(e,e,e,e,something])])]`	*(for) this person to want me to know something*

The non-equi construction has the negative variant

(18) $p(want, [Subject1, p(not(Pred), [Subject2, ...])])$

for "*X wants Y not to do Z*"-type constructions.
Examples:

NSM-Prolog	English NSM

`p(want,[me,`
`  p(not(see), [you,`
`  sp(this,e,e,e,something)`
`  ])`
`])`
	(for) me to want you not to see this thing

`p(want, [sp(this,e,e,e,someone),`
`  p(not(know), [me,`
`  sp(this,e,e,e,something])])}`
	(for) this person to want me not to know this thing

4.3 Predicates with Variable Valences

4.3.1 DO

The predicate *do* selects a frame with five arguments: an agent, an object (DO SOMETHING), a patient (DO SOMETHING TO SOMEONE), an instrument (DO SOMETHING WITH SOMETHING) and a comitative (DO SOMETHING WITH SOMEONE). When the object is modified by an evaluator (`good` or `bad`), it can open a sixth slot in the valence-frame, a *benefactive* (DO SOMETHING GOOD/BAD FOR SOMEONE).

Accodringly, `do` enters the following two frames:

(19) $p(do, [Agent, Object, Patient, Instrument, Comitative])$

and

(20) $p(do, [Agent, Object, Patient, Instrument, Comitative, Benefactive])$

Examples:

NSM-Prolog	English NSM
`p(do, [sp(e,e,e,e,someone), sp(e,e,e,e,something), e,e,e])`	*(for) someone to do something*
`p(do, [me, sp(e,e,e,good,something), e,e,e])`	*(for) me to do something good*
`p(do, [you, sp(e,e,e,good,something), me,e,e])`	*(for) you to do something good to me*
`p(do, [sp(this,e,e,e,someone), sp(e,e,e,e,something), e, sp(this,e,e,e,something), e])`	*(for) this person to do something with this thing*
`p(do, [sp(this,e,e,e,someone), sp(e,e,e,e,something), e,e,me])`	*(for) this person to do something with me*
`p(do, [sp(this,e,e,e,someone), sp(e,e,e,good,something), e,e,e,me])`	*(for) this person to do something good for me*

4.3.2 THINK

The predicate `think` selects four NSM-frames:

1. `SOMEONE THINKS ABOUT SOMEONE/SOMETHING`
   which in NSM-PROLOG becomes:

   (21) $p(think, [Subject, Topic])$

   Examples:

NSM-Prolog	English NSM
`p(think, [me, you])`	*(for) me to think about you*
`p(think, [sp(this,e,e,e,someone), sp(this,e,e,e,something)])`	*(for) this person to think about this thing*
`p(think,[me,e])`	*(for) me to think*

   As we see from the last example, if the object slot is empty, we have the intransitive construction `X THINKS`.

2. `SOMEONE THINKS SOMETHING GOOD/BAD ABOUT SOMEONE`

(22) $p(think, [Subject, Eval, Topic])$

Examples:

NSM-Prolog	English NSM
`p(think, [me, sp(e,e,e,good, something), you])`	*(for) me to think something good about you*
`p(think,[sp(this,e,e,e,someone), sp(e,e,e,very(bad),something), sp(this,e,e,e,something)])`	*for this person to think something very bad about this thing*

3. `SOMEONE THINKS LIKE THIS: ‘‘ _ ’’`

This frame is described by a *Sentence group* structure, which uses PROLOG list, and will be presented in section 7.1.

(23) $p(think, [Subject, SentenceGroup])$.

The frame has a variant:

4. `SOMEONE THINKS LIKE THIS ABOUT SOMETHING: ‘‘ _ ’’`

(24) $p(think, [Subject, SentenceGroup, Topic])$.

The first argument of `think` is a sentence group, the second argument is the topic (which will be rendered into an English-NSM *about*-phrase).

5. `SOMEONE THINKS THAT` *`proposition`*

The predicate `think`, in this case, selects a subject and a *proposition*, represented by one of the two predicates `prop(Sentence)` and `maybe(Sentence)`, which take a *tensed sentence* as their argument.[14] Examples will be found in section 5.

(25) $p(think, [Subject, prop(Sentence)])$

(26) $p(think, [Subject, maybe(S)])$

4.3.3 KNOW

Here are the NSM constructions in which the prime `know` can enter.

1. `X KNOWS SOMETHING`

(27) $p(know, [Subject, Object])$

[14] A *tensed sentence* is an `s`-structure with a time adjunct, that is, with its *Time* slot not empty. Cf. Goddard & Karlsson (2008).

Examples:

NSM-Prolog	English NSM
`p(know,[me, sp(e,e,e,e,something)])`	*(for) me to know something*
`p(know, [sp(this,e,e,e,someone), sp(e,e,all,e,something)])`	*(for) this person to know everything*

2. `X KNOWS SOMETHING ABOUT SOMETHING`

(28) $p(know, [Subject, Object, Topic])$

Examples:

NSM-Prolog	English NSM
`p(know,[me, sp(e,e,e,e,something), sp(this,e,e,e,someone)])`	*(for) me to know something about this person*
`p(know, [you, sp(e,e,e,e,something), me])`	*(for) you to know something about me*

The next to frame have propositional arguments. The direct quotation frame uses a sentence-group construction:

3. `X KNOWS: _`

(29) $p(know, [Subject, SentenceGroup)]).$

The indirect quotation frame uses the same predicate `prop(Sentence)` found in the options for the `think` prime.

4. `X KNOWS THAT` *`proposition`*

(30) $p(know, [Subject, prop(S)])$

To these frames, I added tentatively two *wh*-frames, an "X knows how" frame, represented by the predicate `way(Sentence)`, and a "X knows who (did this)" frame, represented by the predicate `wh(Sentence)`:

5. `X KNOWS HOW` *sentence*

(31) $p(know, [Subject, way(Sentence)]).$

6. `X KNOWS WHO DID THIS`

   (32) $p(know, [Subject, wh(S)]).$

   For instance, to represent the English-NSM sentence *you will know how you can say something to this other person*[15], the *Manner* slot of the sentence predicate `s` is filled by the wh-word `how`, while `wh(Sentence)` fills the second argument of the *ArgList* of the `p` predicate:

NSM-Prolog	English NSM

`s(e,after,e,e,e,`
`  p(know,[you,`
`  wh(`
`  s(e,e,can,e,e,`
`  p(say,[you,`
`  sp(e,e,e,e,something),`
`  sp(this,e,other,e,someone),`
`  e]),`
`  e,how)`
`  )`
`  ]`
`  ),`
`  e,e)`
	you will know how you can say something to this other person

4.3.4 SAY

The valence possibilities for `SAY` are similar to those for `KNOW`. As in the case of `know`, I have added a *wh*-frame.

1. `X SAYS: ‘‘_’’`

   (33) $p(say, [Subject, prop(SentenceGroup)])$

2. `X SAYS THIS TO SOMEONE ABOUT SOMETHING: ‘‘_’’`

   (34) $p(say, [Subject, SentenceGroup, Dative, Topic]).$

3. `X SAYS THIS TO Y: ‘‘_’’`

   (35) $p(say, [Subject, SentenceGroup, Listener])$

4. `X SAYS TO Y (THAT)` *`proposition`*

   (36) $p(say, [Subject, prop(Sentence), Listener]).$

[15] Found in Wierzbicka (2001, p. 184).

5. `X SAYS THIS TO Y ABOUT SOMETHING`

   (37) $p(say, [Subject, Object, Listener, Topic])$.

   The *Listener* and *Topic* slot can be empty[16]. Examples:

NSM-Prolog	English NSM
`p(say,[me, sp(e,e,e,e,something), you, sp(this,e,e,e,someone)])`	*(for) me to say something to you about this person*
`p(say, [you, sp(e,e,e,e,something),me,e])`	*(for) you to say something to me*
`p(say, [you, sp(e,e,e,e,something),e,me])`	*(for) you to say something about me*
`p(say, [you, sp(e,e,e,e,something),e,e])`	*(for) you to say something*

4.3.5 BE

The prime `be` can be used in the following constructions:

1. Specificational BE

   This newly recognised prime[17] is represented as follows:

   (38) $p(be, [Arg1, Arg2])$.

   Example:

NSM-PROLOG	ENGLISH NSM
`p(be, [sp(this,e,e,e,someone), sp(e,e,e,good,someone)])`	*(for) this person (to be) someone good*

2. Existential be (THERE IS)

   The predicate `be` in an unary frame represents the prime THERE IS. The *locus* argument fills the *Location* slot of the sentence frame.

   (39) $p(be, [Arg])$.

   Examples:

[16] As we know, an empty slot is represented by the atom `e`.
[17] See Goddard & Wierzbicka (2008)

NSM-Prolog	English NSM
`p(be,[sp(e,e,e,e,someone)])`	*There is someone*

`s(e,e,e,e,`
`  p(be,[sp(e,many,e,e,people)]),`
`  here).`
	There are many people here

`s(e,e,e,e,`
`  p(be,[sp(e,many,e,e,kind(good,something))]),`
`  e).`
	There are many good kinds of things

3. HAVE

   This prime is tentatively represented with a *mihi est* constructional variant of the previous frame. The *Arg* slot is filled by the possessed, while the possessor is represented by the predicate `poss(SP)`, which goes into the *Location* slot of the sentence frame. The *SP* slot is filled by a `sp`-structure or its equivalents `me` and `you`.

   (40) $p(be, [Arg]).$

   which enters the *PRED* slot of the stative sentence frame:

   (41) $s(Pol, Time, Mod, Dur, PRED, poss(SP))$

   Example:

NSM-Prolog	English NSM

`s(e,e,e,e,`
`  p(be,[sp(e,many,e,e,kind(good,something))]),`
`  poss(me)).}`
	I have many kinds of good things

4. locational BE

   An alternative to the direct representation of predicate phrases with a locational predicate is the following structure:

   (42) $p(be, [Locatum, Location]).$

   where *Location* is filled by a `sp`-structure with `somewhere` as the head noun, or by one of the following PROLOG predicates:

   - `here`
   - `above(SP)`, `below(SP)`
   - `far(SP)`, `near(SP)`

NSM-Prolog	English NSM
`p(be, [sp(this,e,e,e,someone), sp(this,e,e,e,somewhere)]))`	*(for) this person to be in this place*
`p(be, [sp(this,e,e,e,someone), here])).`	*(for) this person not to be here*
`p(be, [sp(this,e,e,e,something), above(me)])).`	*(for) this thing to be above me*
`p(be, [sp(this,e,e,e,something), below(e)])).`	*(for) this thing to be below*

5 Clause Level

5.1 The NSM-PROLOG clause structure

Quoting from Wierzbicka (1996, p. 54),

> The basic unit of the NSM syntax is a "clause", which is constituted by a "substantive" and a "predicate", and some additional elements determined by the nature of the predicate. In addition to this major type of clause (to be discussed below) there is also one minor type, which can be regarded as an analogue of "subjectless sentences" of the traditional grammar, and which includes "existential sentences", cendered on the predicate THERE IS/ARE (e.g. "there are many kinds of birds"; "there is plenty of water here").

This basic clause type can be expanded by various modifiers.
Initially, I distinguished two kinds of clause:

- *Stative* clauses are represented by the six-argument `s` predicate:

  (43) $s(Pol, Time, Mod, Dur, PRED, Loc)$

- *Non stative* clauses are represented by the same `s` predicate, this time taking eight arguments:

  (44) $s(Pol, Time, Mod, Times, Dur, PRED, Loc, Manner)$

  However, I think that it is unnecessary to posit two different `s` structure: stative sentences can be represented by the same functor `s/8` as all the other sentences, leaving empty the $Times$a and $Manner$ slots.

As in the NSM-PROLOG notation the "substantive" referred to by Wierzbicka (the "subject" of the NSM clause) is already present at the phrase level (in the argument-list slot of the `p(Predicate,ArgList)` construction), there is no subject slot in the `s`-structures.

The stative sentence has headed by one of the stative predicates:

- The evaluators and descriptors (`good`, `bad`, `big` and `small`); The locational predicates `above`, `below`, `far`, `near`;
- The predicates `be`, `have`, `live`.

Here are all the slots of the `s` predicate:

1. *Pol*: the Pol(arity) slot is either empty (`e`) or filled by the predicate `not`;
2. the *Time* slot is filled by a time modifier expressing location in time (discussed more fully in section 5.2.1). It can be empty;
3. *Mod*: The Mod(ality) slot is either empty or filled by the prime `can`;
4. The *Times* slot is filled by a *frequency* modifier (section 5.2.3), or empty;
5. The *Dur* slot is filled by a time modifier expressing duration, or left empty;
6. The *PRED* slot is filled by a `p`-structure. It is the only slot of the `s`-structure which cannot be empty;
7. The *Loc* slot contains a location modifier, and can be empty, as can the
8. *Manner* slot, which is filled by a manner modifier.

5.2 The Tense and Aspect Systems

5.2.1 The Time modifier

As we have seen, for each prime which can occupy the *PRED(ICATE)*, as a case of allolexy due to agreement of the predicate with time modifiers. Clauses like *something happened* are in fact a shorthand for *before (this time), something happened*. The deictic component of the English past tense (the fact that the event occurred before the time of speech) is captured in Goddard & Wierzbicka (2002, 69-70) by further analysing *it happened* as

I say this now:
    it happened before

Another possibility could be to consider such forms as `DID` and `IS DOING` as portmanteau for, respectively, `BEFORE + DO` and `NOW + DO`. In this case, the presence of explicit time modifiers would no more be obligatory.

The PROLOG-NSM notation allows for both solutions. The grammar which translates the NSM-PROLOG formulas into language-particular NSM sentences will map the primes which fill the *Time* slot of a clause frame, either onto lexical adverbials, or as tense inflection on the predicate.

The present NSM-PROLOG notation allows for two predicates to fill the *Time* slot of the sentence frame: `t` and `foc`. Both have the same structure, with three arguments:

(45) $t(Location, Reference, Distance)$

I have posited a separated predicate `foc` to distinguish the many cases in which the English tenses are found in NSM literature without a time adverbial. The predicate `t/3` surfaces as tense, while `foc/3` surfaces as tense plus adverbial. Perhaps this distinction is not needed: if English NSM tensed sentences without a time adverbials are just a "shortcut" for a full NSM sentence whose tense is interpreted as an allolex "agreeing" with the time adverbial, then we will be able to discard `foc/3`.

In the following examples, I wil use only `t/3`.

The fillers of the three slots are, in order:

1. *Location*: One of the three atoms `now`, `before`, `after`, or also `some` (AT SOME TIME), `same` (AT THE SAME TIME), `all` (ALWAYS);

2. *Reference* again one of the three atoms `now`, `before`, `after`, but, this time, specifying the *reference point* from which to consider the time location;

3. *distance* the distance of the time location (1) from the reference point (2).

Examples:

- `t(now,e,e)`, `t(before,e,e)`, `t(after,e,e)`: these predicates will be rendered in English NSM by a present past or future tense respectively.

- `t(before,now,e)`, `t(after,now,e)`, are notational variants of `t(before,e,e)` and `t(after,e,e)`, stating explicitly that BEFORE and AFTER take as their reference point the moment of speech (NOW).

- `t(before,now,long)`, `t(before,now,short)`: these formulas represent the English prime combinations A LONG TIME BEFORE, A SHORT TIME BEFORE. In languages which distinguish degrees remoteness in the past, such as Shipibo-Conibo (Faust (1990)), these formulas will generate the corresponding tenses.

- `t(before,after,e)`, `t(after,after,e)`: here time location BEFORE and AFTER is relative to the past (the second BEFORE).

- `t(same,now,e)`, `t(same,before,e)`, `t(same,after,e)`. The prime `same` takes a reference point: AT THE SAME TIME (AS NOW), AT THE SAME TIME (AS BEFORE), AT THE SAME TIME (AS AFTER). These three predicates will all surface as *at the same time* in English NSM, but in tensed languages like English they will trigger, respectively, present, past or future tenses.

- `t(some,now,e)`, `t(some,before,e)`, `t(some,after,e)`. These three structures represent the prime combination AT SOME TIME (that is, at some specific time, Russian *kogda-to*), and in a tensed language will trigger, respectively, present, past or future tenses.

- `t(any,now,e)`, `t(any,before,e)`, `t(any,after,e)`. These three structures represent the prime combination *at some time* (that is, at some unspecified time, Russian *kogda-nibud'*), and in a tensed language will trigger, as in the previous example, respectively present, past or future tenses.

Some examples follow:

NSM-Prolog	English NSM
`t(now,e,e)`	*now*
`t(before,now,e)`	*before*
`t(before,e,e)`	*before*
`t(before,before,e)`	*before* (triggers past perfect)
`t(same,now,e))`	*at the same time* (triggers persent tense)
`t(same,before,e))`	*at the same time* (triggers past tense)
`t(any,e,e)`	*at some time*
`t(any,before,e)`	*at some time* (in the past – triggers past tense)
`t(any,after,e)`	*at some time* (in the future – triggers future tense)
`t(all,e,e)`	*always* (triggers present tense)
`t(all,before,e)`	*always* (triggers past tense)
`t(some,now,e)`	*at some time* (triggers present tense)
`t(some,after,e)`	*at some time* (triggers future tense)
`t(after,before,long)`	*a long time after that time* (triggers past tense)
`t(before,now,short)`	*a short time before* (triggers past tense)

5.2.2 The Time duration modifier

As we have seen, the *Dur* slot is filled by the predicate `dur(Duration)`, where `Duration` can be one of the primes `more`, `not(more)`, `all`, `some`, or the special predicates `long`, and `short`. Examples:

NSM-Prolog	English NSM
`dur(not(more))`	*anymore*
`dur(all)`	*all the time*
`dur(some)`	*for some time*
`dur(long)`	*for a long time*
`dur(short)`	*for a short time*
`dur(some)`	*for some time*

5.2.3 The Time Frequency modifier

The *Freq* slot is filled by the predicate `fr(Frequency)`, where *Frequency* can be one of the primes `one, two, some, many, all`, or the combiation `fr(not(Frequency))`.

Examples:

NSM-Prolog	English NSM
`fr(some)`	*sometimes*
`fr(many)`	*often*
`fr(always)`	*all the times*
`fr(not(all))`	*not all the times*
`fr(one)`	*once*
`fr(two)`	*twice*
`fr(N)`	*N times*

5.2.4 The aspectual operator `i`

In the structure `p(Pred,[Arg1,...])`, the *Pred* slot can also be filled by the predicate `i(Pred)`, where `Pred` is not stative. This feature tentatively adds to the NSM the "internal viewpoint" into a situation, yielding imperfective aspect. It remains to be seen whether NSM needs such a feature. If we imagine the predicate `i` as simply the prime `inside`, then NSM could allow such combinations as BEFORE + INSIDE + DO, which are expressed in English NSM by *(before), he was doing*, and in Polish and Russian NSMs by an imperfective verb in the past tense.

The following two sentence which use progresive aspect in English NSM are taken from Wierzbicka's commentary to Jesus' *logia* and parables (both from the exegesis to Matthew 5, 17-20, Wierzbicka (see 2001, pp. 57-61)):

NSM-Prolog	English NSM

`s(e,before,e,e,dur(long),`
`  p(i(say),`
`  [sp(e,some,e,e,people),`
`  sp(e,e,e,e,something),`
`  sp(this,e,e,e,people),`
`  'God']),`
`  e,e)`
	Some people were saying something to these people about God for a long time

`s(e,foc(now),e,e,e,`
`  p(i(happen),`
`  [sp(this,some,e,e,something),e]),`
`  sp(this,e,e,e,somewhere),e)`
	now these things are happening in this place.[18]

5.3 Location and Manner Modifiers

The *Loc* slot can be filled by an `sp`-structure with the prime `somewhere` in the head position, or by the special primes `here` and `where` (the latter in a wh-sentence).

The *Manner* slot can be filled by the structure `like(this)` or by an `sp` structure with the prime `somehow` as head, or, in a wh-structure, by the prime `how`. Examples:

NSM-Prolog	English NSM

`s(e,before,e,e,e,`
`  p(want,['God',`
`  p(say,[e,`
`  sp(e,e,e,e,something),`
`  sp(e,all,e,e,people),e])`
`  ]),`
`  e,`
`  sp(this,e,e,e,somehow))`
	God wanted to say something to all the people in this way

`s(e,before,e,e,e,`
`  p(do,[me,e,e,e,e,],`
`  like(this),`
`  sp(this,e,e,e,somewhere)).`
	I did like this in this place

6 Sentence Level

In NSM-PROLOG notation, a sentence is

- either an `s` structure (ordinary or stative clause), or
- a complex sentence. Complex sentence are built with ther primes `if`, `when` and `because`, which take two `s`-structures as argument:

(46) $$if(Sentence_1, Sentence_2)$$

(47) $$when(Sentence_1, Sentence_2)$$

(48) $$because(Sentence_1, Sentence_2)$$

(49) $$like(Sentence_1, Sentence_2)$$

Examples now follow of the complex sentence types:

6.1 If

NSM-Prolog	English NSM

`if(s(e,e,e,e,e,`
`  p(do,[you,sp(this,e,e,e,something),e,e,e]),`
`  e,e),`
`  s(e,after,e,e,e,`
`  p(say, [sp(e,e,e,e,people),`
`  sp(e,e,e,bad,something),`
`  e,`
`  you]),`
`  e,e)).`
	If you do this thing, people will say something bad about you

`if(s(e,e,e,e,e,`
`  p(live,[sp(e,e,e,e,man),sp(e,e,e,e,woman)]),`
`  e,e),`
`  s(not,foc(same(now)),can,e,e,`
`  p(live,[sp(this,e,e,e,man),`
`  sp(e,e,other,e,woman)]),`
` e,e))
	If a man lives with a woman, this man cannot live with another woman at the same time

6.2 When

NSM-Prolog	English NSM

`when(`
`  s(e,e,e,e,e,`
`  p(think, [sp(e,e,e,e,man),`
`  sp(e,e,e,like(this),something),`
`  sp(e,e,e,e,woman)]),`
`  e,e),`
`  s(e,e,e,e,e,`
`  p(i(do), [sp(this,e,e,e,man),`
`  sp(e,e,e,bad,something),`
`  sp(this,e,e,e,woman), e,e]),`
`  e,e))`
	when a man thinks something like this about a woman, whis man is doing something bad towards this woman

`when(`
`  s(e,e,e,e,e,`
`  p(think,[you,like(this),e]),`
`  e,e),`
`  s(e,after,e,e,`
`  p(good,[prop(`
`  s(e,e,e,e,e,`
`  p(think,[you,'God']),`
`  e,e)`
`  )])`
`))`
	When you think like this, it is good if you think about God

6.3 Because

NSM-Prolog	English NSM

`because(`
`  s(e,e,e,e,e,`
`  p(know,[sp(e,some,other,e,people),`
`  e,'God']),`
`  e,e),`
`  s(e,e,e,e,e,`
`  p(hear,[top(people),`
`  sp(e,e,e,e,word(me))`
`  ]),`
` e,e))
	other people know God, because they hear my words

`because(`
`  s(e,after,e,e,`
`  p(like,[sp(this,e,e,e,people),`
`  sp(e,one,e,e,someone)])),`
` sp(this,e,e,e,e))
	because of this, these people will be like one person

`if(`
`  s(e,e,e,e,e,`
`  p(do,[sp(e,e,e,e,someone),`
`  sp(e,e,e,bad,something),`
`  sp(e,e,e,e,someone),`
`  e,e]),`
`  e,e),`
`  s(not,after,e,e,`
`  p(bad,[prop(`
`  because(`
`  s(e,e,e,e,e,`
`  p(do,[sp(e,e,e,e,people),`
`  sp(e,e,same,e,something),`
`  sp(this,e,e,e,someone),`
`  e,e]),`
`  e,e),`
`  sp(this,e,e,e,e))`
`  )]`
`  ))`
`)`
	if someone does something bad to someone, it will not be bad if people do the same thing to this person because of this

7 Beyond the Sentence

7.1 Sentence Groups

Textual cohesion is assured in NSM, other than with anaphoric and cataphoric use of primes like THIS and SOMETHING to refer to previous or following sentences, also by the use of *indentation*. Sentences which go together in an unique indentation group form the *sentence group*.

A sentence group is represented as a list of sentences. At first I wanted to add to this list a sort of "introduction" specifying the amount of indentation desired, the options for the labelling of the single sentences of the group, and an optional string; now I am inclined to drop this extra-linguistic information from the repersentation. A sentence group is thus represented by a simple list of sentences.

Examples of texts now follow.

8 Sample texts

8.1 God

Here is the definition of the (Jewish-Christian) concept of God as presented in Wierzbicka (2001, p. 21). I have translated Wierzbicka's text into the following NSM-PROLOG formula:

```
[
s(e,e,e,e,p(be,['God',sp(e,e,e,e,someone)])),
s(not,e,e,e,p(be,['God', sp(e,e,e,e,something)])),
s(e,e,e,e,p(be,[top('God'),sp(e,e,e,good,someone)])),
s(not,e,e,e,p(be,[top('God'),sp(e,e,e,like(people),someone)])),
s(not,e,e,e,p(be,[sp(any,e,other,like(top('God')),someone)]),e),
s(e,e,e,dur(all),p(exist,[top('God')])),
because(s(e,e,e,e,p(exist,[sp(e,all,e,e,something)])),
        s(e,e,e,e,e,p(want,[top('God'),
                                    p(exist,[sp(e,all,e,e,something)])]),e,e)),
because(s(e,e,e,e,p(exist,[sp(e,e,e,e,people)])),
        s(e,e,e,e,e,p(want,
                              [top('God'),
                               p(exist,[sp(e,e,e,e,people)])]),e,e)),
because(s(e,e,e,e,p(exist,[top('God')])),
s(e,e,e,e,p(exist,[top('God')]))),
because(not,
s(e,e,e,e,p(exist,[top('God')])),
sp(any,e,other,e,something)),
s(e,e,e,e,e,p(live,[top('God'),e]),e,e)
].
```

The predicate `top(SP)` replaces an SP by its topicalized version (an English pronoun). This feature is already beyond a strict NSM. The name `'God'` is here treated as a substantive phrase, like `me` and `you`.

The NSM generator program, after it has loaded the English-NSM grammar, outputs the following translation of the preceding formula[19]:

```
(a) God is someone
(b) God isn't something
(c) he is someone good
(d) he isn't someone like people
(e) there isn't anyone else like him
(f) he exists always
(g) everything exists because he wants everything to exist
(h) people exist because he wants people to exist
(i) he exists because he exists
(j) he exists not because of anything else
(k) he lives
```

8.2 Luke 12, 33-34

Here is the Gospel text, quoted from the World English Bible[20]:

> [33] Sell that which you have, and give gifts to the needy. Make for yourselves purses which don't grow old, a treasure in the heavens that doesn't fail, where no thief approaches, neither moth destroys. [34] For where your treasure is, there will your heart be also.

This is Wierzbicka's exegesis translated into NSM-PROLOG:

```
[
s(e,all,e,e,e,p(think,[sp(e,e,e,many,people),
sg("",4,a) ::
   [
   s(e,e,e,e,e,p(want,[me,p(happen,[sp(e,many,e,good,something),me])]),e,e),
   if(
      s(e,e,e,e,e,p(happen,[sp(this,some,e,e,something),e]),e,e),
      s(e,after,e,e,e,p(feel,[me,sp(e,e,e,very(good),something),e]),e,e)),
   because(
      s(e,e,e,e,e,p(want,[me,p(do,[e,sp(e,many,e,e,something),e,e,e])]),e,e),
      sp(this,e,e,e,e))
  ]
   ]),e,e),
if(
  s(e,e,e,e,e,p(think,[you,like(this),e]),e,e),
  s(e,after,e,e,p(good,[prop(
    s(e,foc(same(now)),e,e,e,p(think,[you,sp(e,e,other,e,something),e]),e,e)
  )])))),
```

[19] If you check this program, you will notice some slight differences in the formulas from what is described in this article.

[20] Rainbow Missions (1997-) (World English Bible) is a modern, public-domain English translation of the Bible.

```
s(e,e,e,e,p(good,[prop(
  s(e,e,e,e,e,p(think,[you,sg("",4,a) :: [
    s(e,any(after),e,e,e,p(die,[me]),e,e),
    before(
      s(e,e,can,e,p(good,[sp(this,some,e,e,something),me])),
      s(e,e,e,e,e,p(die,[me]),e,e)),
    after(
      s(not,after,e,e,p(good,[sp(this,some,e,e,something),me])),
      s(e,e,e,e,e,p(die,[me]),e,e))
   ]]),e,e)
  )])),
s(e,all,e,e,p(good,[prop(
 because(not,
       s(e,e,e,e,e,
             p(want,[you,
                 p(do,[e,sp(e,e,e,good,something),e,e,e,
                       sp(e,e,other,e,people)])]),e,e),
       s(e,e,e,e,e,p(think,[you,prop(
             because(
                 s(e,after,e,e,e,p(happen,[sp(e,e,e,good,something),you]),e,e),
                 sp(this,e,e,e,e))
          )]),e,e))
    ),you])),
like(
   s(e,e,e,e,e,p(want,['God',p(want,[you,p(do,[e,sp(e,e,e,good,something),e,e,e,
             sp(e,e,other,e,people)])])]),e,e),
   s(e,e,e,e,e,p(want,['God',
         p(do,[e,sp(e,e,e,good,something),e,e,e,sp(e,all,e,e,people)])]),e,e
         )),
if(
  s(e,e,e,e,e,p(want,[you,
                      p(do,[e,sp(e,e,e,good,something),
                        e,e,e,sp(e,e,other,e,people)])
                     ]),e,e),
  s(e,e,can,e,e,p(live,[you,'God']),e,e)),
s(e,all,e,e,p(good,[prop(
       s(e,e,e,e,e,p(live,[you,'God']),e,e)),you]))].
```

The software translates this formula as follows:

```
[Wierzbicka 2001:159-60]
(a) many people always think like this
(b)     I want many good things to happen to me
(c)     if these things happen , I will feel something very good
(d)     I want to do many things because of this
(e) if you think like this , it will be good if at the same time you
        think something else
```

```
(f) it is good if you think like this
(g)     at some time I will die
(h)     these things can be good for me before I die
(i)     these things will not be good for me after I die
(j) it always is good for you if you want to do something good for other
        people not because you think that something good will happen
        to you because of this
(k) God wants you to want to do something good for other people like
        God wants to do something good for all people
(l) if you want to do something good for other people , you can live
        with God
(m) it always is good for you if you live with God
```

---

9 Conclusion

Of course, the NSM-PROLOG notation and the NSM-PROLOG grammars are just a first step. Many things remain to be done:

- First of all, the notation has still some inconsistencies (such as the double representation of some structures, like locational predicates and possession), which will have to be polished (unless they will turn out to be justified).
- Some primes are allowed by NSM-PROLOG to occur in more than one role: for example, `like` can be attribute in A PERSON LIKE ME and predicate in YOU ARE LIKE ME. It remains to be checked whether all these uses are indeed universal, or whether just one of them must be selected for "strict" NSM, while leaving the alternative to the "syntactic shortcuts".
- In developing other language particular NSM grammars, maybe additional features will have to be added to the NSM-PROLOG notation. On the other hand, some distinction made in this document could turn out to be outside a strict NSM, though they are used in NSM texts. In this way, NSM-PROLOG notation can help to define a "standard specification" for NSM. While NSM texts can use various shortcuts in order to come closer to idiomatic language, texts to be used used by NLP tools should adhere strictly to the specification;
- though it is possible to write formulas "by hand" and then feed them to a program for text generation, the best use of NSM-PROLOG in NLP applications is probably that of an intermediate metalanguage between a parser and a generator, in automatic translation. A PROLOG parser-generator which is being developed which uses NSM formulas in this way.

References

Faust, Norma. 1990. *Lecciones para el aprentizaje del idioma Shipibo-Conibo.* Instituto Linguístico de Verano.

Goddard, Cliff. 2002. The On-going Development of the NSM Research Program. vol. gowie2:02. Amsterdam: Benjamins.

Goddard, Cliff (ed). 2006. *Ethnopragmatics. Understanding Discourse in Cultural Context.* Mouton de Gruyter.

Goddard, Cliff (ed). 2008a. *Cross-Linguistic Semantics.* John Benjamins.

Goddard, Cliff. 2008b. Natural Semantic Metalanguage : The state of the art. *Pages 1–34 of: Cross-Linguistic Semantics.*

Goddard, Cliff, & Karlsson, Susanna. 2008. Re-thinking THINK in contrastive perspective: Swedish vs. English. *Pages 225–240 of: Cross-Linguistic Semantics.*

Goddard, Cliff, & Wierzbicka, Anna. 2002. *Meaning and universal grammar : theory and empirical findings.* Vol. 1. Amsterdam: Benjamins.

Goddard, Cliff, & Wierzbicka, Anna. 2004. Special issue on Cultural scripts. *Intercultural Pragmatics 1(2).*

Goddard, Cliff, & Wierzbicka, Anna. 2008. New semantic primes and new syntactic frames : "Specificational BE" and "abstract THIS/IT". *Pages 35–57 of: Cross-Linguistic Semantics.*

Rainbow Missions, Inc. 1997-. *The World English Bible.* URL: http://www.ebible.org.

Wierzbicka, Anna. 1987. *English Speech Act Verbs. A Semantic dictionary.* Academic Press Australia.

Wierzbicka, Anna. 1996. *Semantics: Primes and Universals.* Oxford Univ. Press.

Wierzbicka, Anna. 2001. *What did Jesus mean?* Oxford University Press.

Wierzbicka, Anna. 2002a. Russian cultural scripts: The theory of cultural scripts and its applications. *Ethos 30(4)*, 401–432. URL: http://www.une.edu.au/lcl/nsm/pdf/AW_Russian_Cultural_Scripts.pdf.

Wierzbicka, Anna. 2002b. Semantic Primes and Linguistic Typology. *Pages 225–261 of: Meaning and universal grammar : theory and empirical findings.*

Zamblera, Francesco. 2010a. The English NSM generator grammar. Draft.

Zamblera, Francesco. 2010b. Hist-morph, a PROLOG tool for historical morphology. Draft.