Language-Neutral Syntax: An Overview

Richard Campbell
Hisami Suzuki

July 2002

Technical Report
MSR-TR-2002-76

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
{richcamp,hisamis}@microsoft.com

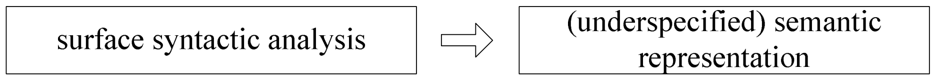
Table of contents

1 Motivation

The aim of this paper is to describe Language-Neutral Syntax (LNS), a system of representation for natural language sentences that is semantically motivated and abstract, yet sufficiently concrete to mediate effectively between languages and between applications in a robust manner. LNS is currently implemented as the output of the NLPWin system under development at Microsoft Research (Heidorn, 2000), but in principle can be output by any system for any language. The survey of LNS provided here is fairly comprehensive; a more selective overview of the basic properties of LNS can be found in Campbell and Suzuki (2002).

Natural language understanding (NLU) systems often make use of a level of semantic or quasi-semantic representation, derived from a surface-based syntactic analysis:

Typical NLU system:


surface syntactic analysis
(underspecified) semantic representation

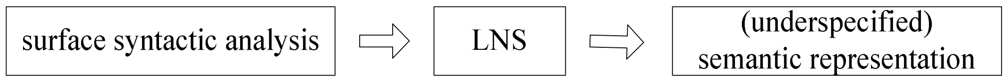
Examples of such semantic levels include Quasi Logical Form (Alshawi *et al*., 1991), Underspecified Discourse Representation Structures (Reyle, 1993), Language for Underspecified Discourse representations (Bos, 1995) and Minimal Recursion Semantics (Copestake *et al*., 1995, 1999). In systems of this sort, the semantic level usually serves to mediate between languages in multilingual applications such as semantic-transfer-based machine translation (MT).

While these semantic representations are clearly useful and desirable, it is often difficult in practice, and unnecessary for most applications, to have a fully articulated logical/semantic representation. Consider the ADJ+NOUN combinations *black cat* and *legal problem*; both have exactly the same structure, yet the semantic relation between the adjective and noun is different in the two cases: the first is interpreted as $\lambda x[\mathit{black}(x) \wedge \mathit{cat}(x)]$; i.e., a cat which is black; while the second is not normally interpreted as $\lambda x[\mathit{legal}(x) \wedge \mathit{problem}(x)]$; i.e., a problem which is legal. The reason is that, while *black* is an intersecting adjective in the sense of Keenan and Faltz (1985), *legal* is not (or need not be), especially when combined with certain nouns. A fully articulated logical representation would have to at least distinguish intersecting from non-intersecting adjectives to adequately treat both cases. In an NLU context, this would in turn entail extensive lexical annotation, indicating how each adjective sense modifies a noun, if not how each ADJ+NOUN combination has to be interpreted; see for example the summary in Bouillon and Viegas (1999) (note that calling *legal problem* a compound only renames the problem, but doesn't solve it). A system that requires such detailed lexical information would most likely be extremely brittle in the face of a realistically broad range of input; e.g. any input that includes such phrases as *legal problem* and *black bear*.

For the vast majority of applications, however, it is not necessary to make this distinction. For example, in transfer-based machine translation (MT), all we would need to know for the vast majority of ADJ+NOUN combinations is that the adjective modifies the noun; thus we could translate *black cat* to French *chat noir* lit. 'cat black' and *legal problem* to Fr. *problème légal* lit. 'problem legal' without knowing the exact truth-functional relation between adjective and noun. This more basic structural information is the kind of representation that LNS provides.

LNS thus occupies a middle ground between surface-based syntax and full-fledged semantics, being neither a comprehensive semantic representation, nor a syntactic analysis of a particular language, but a semantically motivated, language-neutral syntactic representation.

NLU with LNS:



LNS represents the logical arrangement of the parts of a sentence, independent of arbitrary, language-particular aspects of structure such as word order, inflectional morphology, function words, etc. Thus *black cat* and *legal problem* have the same LNS structure, despite their deep semantic differences, and *black cat* has the same LNS structure as *chat noir*, despite their superficial syntactic differences (see Section 2.3). These two somewhat conflicting requirements of LNS are summarized in the following design criteria of LNS:

LNS design criteria:

1. LNS must be abstract enough to be language-neutral; i.e., to allow deeper, possibly application-specific, semantic representations to be derived from it by language-independent functions.
2. LNS must preserve potentially meaningful surface distinctions; i.e., surface distinctions must be recoverable from LNS.

What characterizes LNS is the particular balance we tried to strike between these two requirements; Section 3 and 4 give a detailed description of these aspects of LNS.

The paper is organized as follows: Sections 2 and 3 are about the structure of LNS representations: Section 2 sketches the formal structure of LNS (supplemented by the tables in the Appendix), and Section 3 discusses the LNS analysis of various linguistic phenomena, with an emphasis on language-neutrality. The next two sections discuss the relationship between LNS and other representations: Section 4 is concerned with the relation between LNS and surface syntax, while Section 5 is about deriving semantic representations from LNS. In Section 6 LNS is compared to other representational frameworks, and Section 7 offers a conclusion.

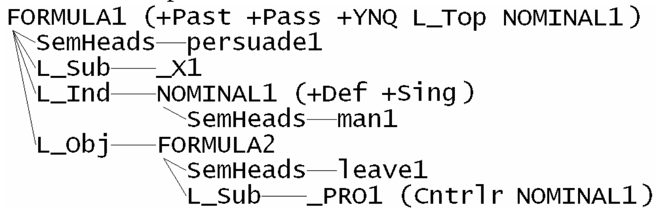
2 LNS structure

The LNS of a sentence is an annotated tree (i.e., each node has at most one parent), but differs from surface-syntax trees in that constituents are not ordered, and in that the immediate constituents of a given node are identified by labeled arcs indicating a semantically motivated relation to the parent node. LNS is thus a combination of constituent structure and dependency structure. An LNS tree is fully specified by defining a dominance relation among the nodes, and specifying the attributes (incl. relations to other nodes) and features of each node. The Appendix contains a description of the basic attributes and features currently used in LNS; in the main body of text we present attributes and features as needed.

2.1 Overview

The basic structure of LNS is best illustrated by looking at an example; the LNS for the sentence *Was the man persuaded to leave?* is given below:[1]

(1) Was the man persuaded to leave?



```
FORMULA1 (+Past +Pass +YNQ L_Top NOMINAL1)
 \SemHeads—persuade1
 \L_Sub———_X1
 \L_Ind———NOMINAL1 (+Def +Sing)
           \SemHeads—man1
 \L_Obj———FORMULA2
           \SemHeads—leave1
           \L_Sub———_PRO1 (Cntrlr NOMINAL1)
```

Non-terminal nodes have either NOMINAL or FORMULA as a nodetype, while terminal nodes are lexemes in a given language (or abstract expressions such as variables; see below). Non-terminal nodes correspond roughly to the phrasal and sentential nodes of traditional syntactic trees. We adopt the convention that each non-terminal node is either the root of the tree, the value of a labeled arc other than *SemHeads* (or semantic head, discussed below), the value of some other attribute (such as *Cntrlr*, see below), or has multiple branches. This convention reflects no linguistic principle, but is merely a convention to avoid unnecessary proliferation of nodes.

The labeled arcs in the tree represent "deep" grammatical functions, or GFs (logical subject, logical object, etc.), and other semantically motivated relations such as SemHeads. These are the attributes that constitute the LNS tree, and are henceforth referred to as *tree attributes*. In this passive example, the logical subject (*L_Sub*) is unspecified, the logical object or complement (*L_Obj*) is the subordinate clause, and the logical indirect object (*L_Ind*) is the surface subject.

The fact that *the man* is the surface subject is recorded indirectly in the *L_Top* (logical topic) attribute of the root node FORMULA1. L_Top differs from tree attributes like L_Sub and SemHeads that are displayed as labeled arcs in the tree in that it is not part of the tree per se, but is considered an annotation of the tree. Another such *non-tree attribute* in (1) is Cntrlr, an

[1] LNS structures displayed in this paper are snapshots of those generated by the NLPWin system; the root node is in the upper left, and features and non-tree attributes are displayed within parentheses to the right of each node. These and other display conventions are part of NLPWin and not LNS per se.

attribute of certain expressions, like *_PRO*, relative pronouns, etc., which behave semantically as bound variables or otherwise derive their reference from another node; in this case, the Cntrlr of *_PRO1* is NOMINAL1 in the Equi construction. Non-tree attributes tend to indicate non-local dependencies, while tree attributes indicate underlying GFs. For purposes of illustration, non-tree attributes are not displayed as labeled arcs, but either as distinct annotations, as in (1), or not at all, if they are not relevant to the discussion.

Another important feature of LNS is that content words are lemmatized, while function words, such as the definite article, and inflectional morphology, such as the tense and voice of *was persuaded*, are omitted altogether, often replaced by features (*+Def* and {*+Past +Pass*}, respectively, in this example); FORMULA1 is also *+YNQ*, indicating that it is a yes/no question; see Section 2.5, below. Words that are analyzed as not contributing any lexical meaning at all, such as pleonastic pronouns and the copula (see Section 3.4), have no LNS node.

Linear order of constituents in surface syntax is often meaningful; for example, order (combined with voice and case-marking) is one way that deep GFs are marked on the surface (see Section 3.1). LNS constituents are not ordered, however; the information conveyed by order in a particular language is represented more transparently in LNS. To take a simple example, the fact that *the man* precedes the verb in (1) indicates that it is the surface subject, which combined with the passive morphology indirectly indicates that it is the logical indirect object; the LNS represents the latter directly by making it the L_Ind.

A final feature of LNS on display in (1) is the use of expressions which are neither in the surface string, nor lemmas (citation forms) of surface-string expressions. In this example, *_X* is the L_Sub of FORMULA1, indicating that the agent of *persuade* is unspecified; the L_Sub of FORMULA2 is *_PRO*, a controlled expression as described above. Other abstract expressions appear in examples below.[2]

To sum up this section, an LNS is an unordered tree, with labeled arcs (tree attributes) indicating semantic roles, and annotated with features and non-tree attributes.

2.2 SemHeads

SemHeads identifies the semantic head or heads of a constituent; two major points need to be mentioned regarding this attribute: First, SemHeads does not always correspond to the surface-syntactic head; a good illustration is provided by a negative sentence:

(2) He didn't die.

```
FORMULA1 (+Past +Proposition)
 \SemHeads—not1
  \OpDomain—FORMULA2
             \SemHeads—die1
              \L_Sub——NOMINAL1
                        \SemHeads—he1
```

Negation is discussed in more detail in Section 3; for now, it is sufficient to note that a negative sentence has a negative operator (*not* in this example) in SemHeads, taking the kernel sentence in its scope (indicated by *OpDomain*). Although in most theories of syntax *not* is not the surface-syntactic head of this sentence, in LNS sentence-level logical operators like negation are analyzed as SemHeads.

[2] The only important difference between *_PRO* and *_X* is that one is controlled and the other is not; it is therefore not strictly necessary for them to have distinct lemmas.

Second, there may be more than one SemHeads for a given constituent. This occurs in coordinate structures, as shown here; *L_Crd* indicates the coordinating conjunction, if there is one:[3]

(3) Tom, Mary and me

```
NOMINAL1
 SemHeads—NOMINAL2
            SemHeads—Tom1
          NOMINAL3
            SemHeads—Mary1
          NOMINAL4
            SemHeads—I1
 L_Crd—and1
```

In this example, NOMINAL1 has three SemHeads, corresponding to the three conjuncts in the coordinate structure.

Note that the value of SemHeads could itself be a coordinate structure, giving rise to a hierarchical arrangement of conjuncts:

(4) Tom and either Mary or me

```
NOMINAL1
 SemHeads—NOMINAL2
            SemHeads—Tom1
          NOMINAL3
            SemHeads—NOMINAL4
                       SemHeads—Mary1
                     NOMINAL5
                       SemHeads—I1
            L_Mods—either1
            L_Crd—or1
 L_Crd—and1
```

In this example, *either* marks the scope of disjunction, which is therefore narrower than the scope of *and*.

2.3 Scope of operators and modifiers

As noted above with respect to (2), sentence-level operators are assigned to SemHeads in LNS.[4] The operand is either in OpDomain, as in (2), or in *ModalDomain*, if the operator is a modal verb (the motivation for distinguishing OpDomain and ModalDomain is simply to facilitate recovery of the information that the operator is a modal in (5) but not in (2); there is no strictly semantic motivation for the distinction):

(5) You must leave now.

```
FORMULA1 (+Pres +Proposition)
 SemHeads—must1
 ModalDomain—FORMULA2
               SemHeads—leave1
               L_Sub—NOMINAL1
                       SemHeads—you1
               L_Time—FORMULA3
                       SemHeads—now1
```

[3] For languages with coordinate structures without coordinating conjunctions, such as Chinese or Japanese VP coordination, there need not be an L_Crd.
[4] Quantified NPs are not currently assigned scope in LNS; see also Note 15.

The purpose of this analysis is to make the scope of operators explicit in LNS: the scope of each operator is just its OpDomain or ModalDomain. Below is an example with multiple sentence-level operators:

(6) He didn't just die.

```
FORMULA1 (+Past +Proposition)
 \SemHeads—not1
  \OpDomain—FORMULA2
             \SemHeads—just1
              \OpDomain—FORMULA3
                          \SemHeads—die1
                           \L_Sub——NOMINAL1
                                     \SemHeads—he1
```

Here *not* has wider scope than *just*; this is realized in English by linear order, so reversing the order of the modifiers in the English sentence results in a different LNS, with different scope assignments to the operators:

(7) He just didn't die.

```
FORMULA1 (+Past +Proposition)
 \SemHeads—just1
  \OpDomain—FORMULA2
             \SemHeads—not1
              \OpDomain—FORMULA3
                          \SemHeads—die1
                           \L_Sub——NOMINAL1
                                     \SemHeads—he1
```

The scope of modifiers is similarly represented, but the modifier is not assigned to SemHeads, but to some other GF relation; below is an example of a noun phrase with multiple attributive adjectives:

(8) the heaviest natural isotope

```
NOMINAL1 (+Def +Sing)
 \SemHeads—NOMINAL2
 |           \SemHeads—isotope1
 |            \L_Attrib—FORMULA1
 |                        \SemHeads—natural1
  \L_Attrib—FORMULA2 (+Supr +PosSupr)
             \SemHeads—heavy1
```

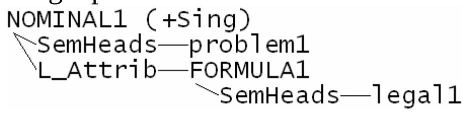
The superlative *heaviest* modifies the ADJ+NOUN combination *natural isotope*, and is represented in LNS as the logical attribute (*L_Attrib*) of NOMINAL1, modifying NOMINAL2. The representation of modifier scope is taken up in Section 3.2 below.

Note that the relation L_Attrib is used for non-quantificational modifiers in NP, regardless of the semantic type of modification. Thus *black cat* and *legal problem*, despite the semantic difference in the type of modification, have the same LNS structure:

(9) a black cat

```
NOMINAL1 (+Sing)
 \SemHeads—cat1
  \L_Attrib—FORMULA1
              \SemHeads—black1
```

(10) a legal problem



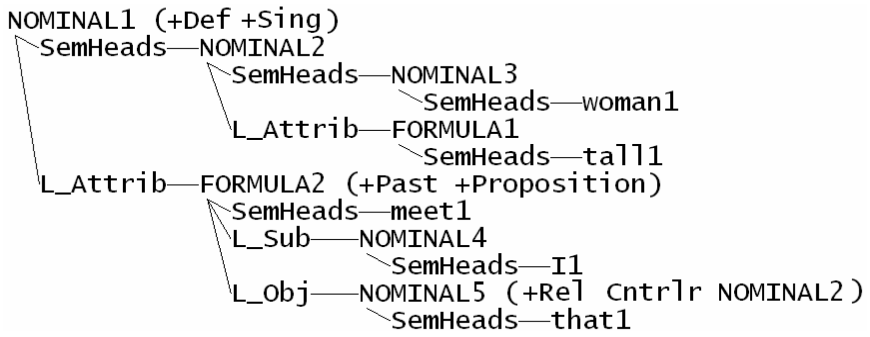
```
NOMINAL1 (+Sing)
 \SemHeads—problem1
  L_Attrib—FORMULA1
                \SemHeads—legal1
```

Thus as discussed in Section 1, LNS provides a fairly shallow representation of this construction, and avoids the brittleness problem posed by extensive lexical annotation of adjectives.

2.4 Variables and Cntrlr

In addition to Equi constructions such as (1), the Cntrlr attribute is used to link relative pronouns to their antecedents in relative clauses and clefts (including both it-clefts and pseudoclefts); (11) shows a simple relative clause:

(11) the tall woman that I met

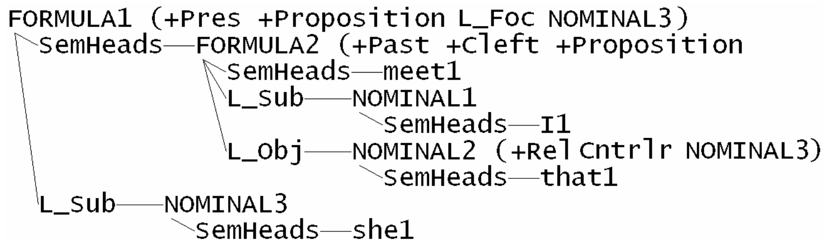


```
NOMINAL1 (+Def +Sing)
 \SemHeads—NOMINAL2
  |           \SemHeads—NOMINAL3
  |            |          \SemHeads—woman1
  |            L_Attrib—FORMULA1
  |                       \SemHeads—tall1
  L_Attrib—FORMULA2 (+Past +Proposition)
              \SemHeads—meet1
               L_Sub——NOMINAL4
               |        \SemHeads—I1
               L_Obj——NOMINAL5 (+Rel Cntrlr NOMINAL2)
                        \SemHeads—that1
```

The relative pronoun NOMINAL3 is controlled by NOMINAL2 in this example, but is not replaced by it, nor by its copy; instead, it is treated as a variable bound by its Cntrlr, hence free in FORMULA1. FORMULA1 is therefore an open sentence, the interpretation of which is λx[*meet*(*I*,x)] (ignoring tense).

Relative pronouns in clefts (both it-clefts and pseudoclefts) work essentially the same way, as in the following example (the non-tree attribute *L_Foc* indicates the focus of the cleft):

(12) It's her that I met.



```
FORMULA1 (+Pres +Proposition L_Foc NOMINAL3)
 \SemHeads—FORMULA2 (+Past +Cleft +Proposition
  |           \SemHeads—meet1
  |            L_Sub——NOMINAL1
  |            |        \SemHeads—I1
  |            L_Obj——NOMINAL2 (+Rel Cntrlr NOMINAL3)
  |                     \SemHeads—that1
  L_Sub——NOMINAL3
           \SemHeads—she1
```

NOMINAL2 acts as a variable, as in the relative clause example above, so that again FORMULA2 is an open sentence, interpreted as λx[*meet*(*I*,x)]. The presupposition of the cleft, in this case that I met someone, can then be obtained by existential closure over FORMULA2 (see also Section 5, below).

2.5 Clause type

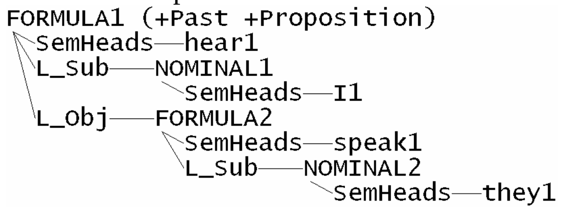
LNS uses features on the clausal constituent to distinguish questions, declaratives and imperatives. The full set of features for clause type is *YNQ* for yes/no questions, *WhQ* for wh-questions, *Imper* for imperatives and *Proposition* for declaratives. These features can occur in root clauses or embedded clauses, and can occur on full or reduced clauses or (in the case of Proposition) small clauses.

For example, the italicized complement clause is +Proposition in each of the following examples:

(13) I believe *he is smart*.
(14) I consider *him to be smart*.
(15) I consider *him smart*.

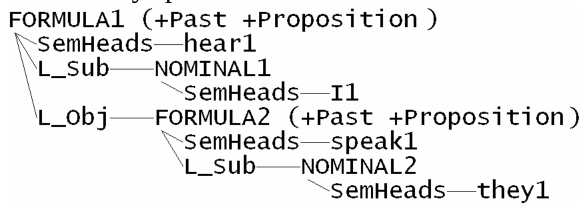
+Proposition indicates that the LNS constituent has a truth value that somehow contributes to the semantic composition of the whole sentence; e.g. in each of these examples what is believed or considered is the truth value of the proposition 'he is smart'. In contrast, the English bare infinitive complement of a perception verb is not +Proposition; the following sentences contrast minimally in LNS:

(16) I heard them speak.



```
FORMULA1 (+Past +Proposition)
 \SemHeads—hear1
 \L_Sub——NOMINAL1
  \          \SemHeads—I1
   L_Obj——FORMULA2
              \SemHeads—speak1
               L_Sub——NOMINAL2
                         \SemHeads—they1
```

(17) I heard they spoke.

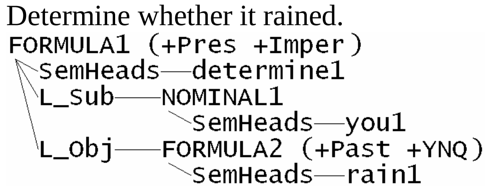


```
FORMULA1 (+Past +Proposition )
 \SemHeads—hear1
 \L_Sub——NOMINAL1
  \          \SemHeads—I1
   L_Obj——FORMULA2 (+Past +Proposition)
              \SemHeads—speak1
               L_Sub——NOMINAL2
                         \SemHeads—they1
```

These LNSs differ only in the features of FORMULA2, which is +Proposition (and also +Past) in (17), but not in (16), reflecting the fact that what is heard is an event in (16), but a proposition with a truth value in (17).

The clause-type feature of a node is not an indication of its semantic composition, but rather of what kind of semantic object it contributes to its semantic or discourse environment: its function in the matrix (for embedded clauses) or its direct speech act type (for root clauses). For example, it may be useful to think of a yes/no question as consisting of a yes/no question operator and a propositional operand, this composition is not reflected in LNS, however; instead, the yes/no question is marked +YNQ, and not +Proposition:

(18) Determine whether it rained.



```
FORMULA1 (+Pres +Imper)
 \SemHeads—determine1
 \L_Sub——NOMINAL1
  \          \SemHeads—you1
   L_Obj——FORMULA2 (+Past +YNQ)
              \SemHeads—rain1
```

Whatever its semantic composition, the contribution of FORMULA2 to the meaning of FORMULA1 is as the question whose answer is to be determined.

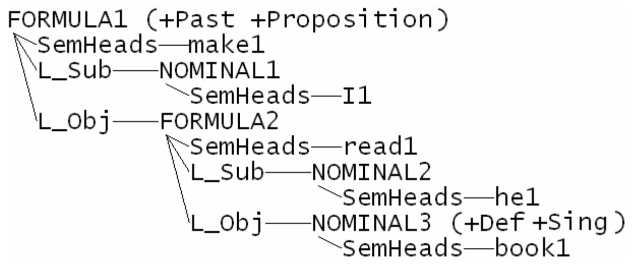
3 Normalization of cross-linguistic variation

In this section the LNS analyses of various linguistic phenomena are presented and discussed. The emphasis here is on the language-neutrality of LNS; specifically, how surface morpho-syntactic variation across languages is normalized for structurally equivalent sentences. As noted in Section 1, LNS is situated between the level of language-particular syntax and various possibly application-specific semantic representations. Though exactly what is normalized is a matter of degree, a higher degree of language-neutrality is generally desirable, not only in principle but also in facilitating multi-lingual applications such as MT. This is the motivation for the first design criterion for LNS in Section 1. In this section we discuss the LNS representation of grammatical relations, modifier scope, sentential negation and copular constructions.

3.1 Grammatical relations

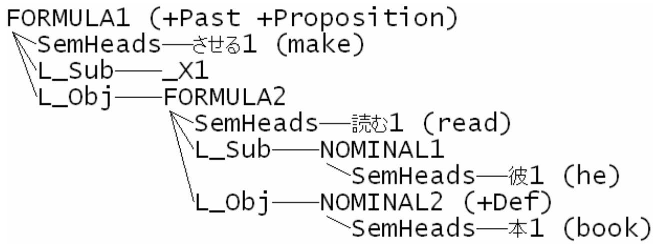
We have already seen how deep grammatical relations are represented in LNS. This subsection is concerned with the normalization of such relations across languages, even when the languages use very different surface encodings of this information. Grammatical relations can be encoded by using various morpho-syntactic devices across languages, such as word order, inflectional or agglutinative morphology, function words, or by the combination of the above. For example, in causative construction, English (19) and Japanese (20) use very different strategies for encoding grammatical relations:

(19) I made him read the book.



```
FORMULA1 (+Past +Proposition)
 \SemHeads──make1
  \L_Sub───NOMINAL1
  |         \SemHeads──I1
  \L_Obj───FORMULA2
             \SemHeads──read1
              \L_Sub───NOMINAL2
              |         \SemHeads──he1
              \L_Obj───NOMINAL3 (+Def +Sing)
                        \SemHeads──book1
```

(20) 彼にその本を読ませた。
kare-ni sono hon-o yoma-se-ta
he-DAT the book-ACC read-CAUS-PAST

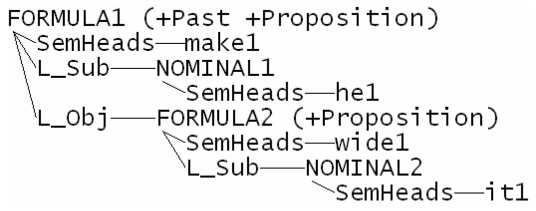


```
FORMULA1 (+Past +Proposition)
 \SemHeads──させる1 (make)
  \L_Sub───_X1
  \L_Obj───FORMULA2
             \SemHeads──読む1 (read)
              \L_Sub───NOMINAL1
              |         \SemHeads──彼1 (he)
              \L_Obj───NOMINAL2 (+Def)
                        \SemHeads──本1 (book)
```

In English (19), it is primarily constituent order (and marginally case in the case of the pronoun *him*) that encodes deep grammatical relations, along with the main verb *made*, which signals the causative construction. In Japanese (20), in contrast, word order does not signify grammatical relations; instead, they are indicated by various case particles: nominative が *ga*, accusative を *o* and dative に *ni*. The causative construction is indicated by the bound morpheme *se* agglutinated to the main verb stem *yoma* 'read'. The sentence is therefore monoclausal in its surface syntactic structure, with three arguments indicated by three different case markers. At LNS, however, two predicates are identified, the causative predicate *saseru* on the one hand, and *yomu* 'read' on the other, each of which has its own subject and object. Note that this LNS structure is motivated on Japanese internal grounds: there is no way of fitting the NPs into correct grammatical relations on the basis of the mono-clausal surface syntactic structure. That is to say, LNS represents what the language-particular syntactic structure expresses using language-neutral vocabulary, such as grammatical relations (L_Sub and L_Obj in the above examples); often, this also means normalizing structurally equivalent sentences across languages into a shared representation, as in the case of (19) and (20) above.

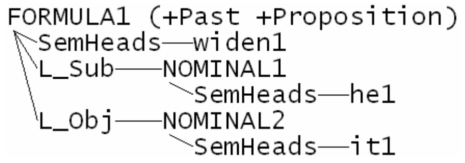
LNS normalizes structurally equivalent expressions across languages; it does not simply normalize expressions that mean the same thing. For example, sentences in (21) and (22) below mean roughly the same thing, yet the LNS structures are distinct:

(21) He made it wide.



```
FORMULA1 (+Past +Proposition)
 SemHeads—make1
 L_Sub——NOMINAL1
            SemHeads—he1
 L_Obj——FORMULA2 (+Proposition)
            SemHeads—wide1
            L_Sub——NOMINAL2
                      SemHeads—it1
```

(22) He widened it.



```
FORMULA1 (+Past +Proposition)
 SemHeads—widen1
 L_Sub——NOMINAL1
            SemHeads—he1
 L_Obj——NOMINAL2
            SemHeads—it1
```

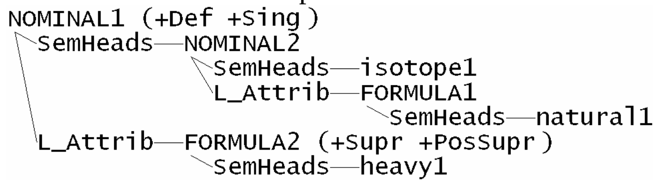
(21) is an analytic causative construction, while (22) expresses similar content using the lexical causative *widen*. There is ample evidence that (22) is not bi-clausal, in contrast to the analytic causative constructions as in (19) and (20): for example, only analytic causatives allow extra arguments. LNS normalizes what can be normalized on the basis of morpho-syntactic evidence; it does not perform lexical decomposition or generative semantics. Therefore, LNS is not an interlingua (in the sense that it is not a semantic representation); rather, it is a language-neutral representation of syntactic structure.

3.2 Modifier scope

The order of modifiers within NP is a domain that shows a great deal of cross-linguistic variation, and therefore makes for a good demonstration of the language-neutrality of LNS. As elsewhere, the discussion here focuses on the structure of LNS, not on its computation; the reader is referred to Campbell (2002) for discussion of the computation of modifier scope in NP.

Consider again (8), repeated here:

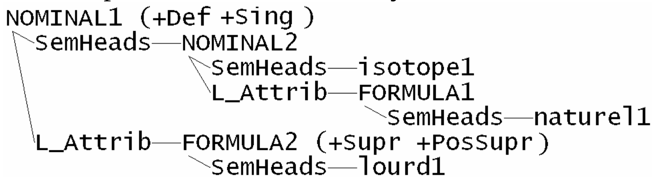
(8) the heaviest natural isotope



```
NOMINAL1 (+Def +Sing)
 SemHeads—NOMINAL2
             SemHeads—isotope1
             L_Attrib—FORMULA1
                        SemHeads—natural1
 L_Attrib—FORMULA2 (+Supr +PosSupr)
             SemHeads—heavy1
```

As noted earlier, *heavy* has wider scope than *natural*; this is realized in English surface syntax as L-R linear order. In French, however, both modifiers can be postnominal, with the order reversed:

(23) l' isotope naturel le plus lourd
the isotope natural the most heavy

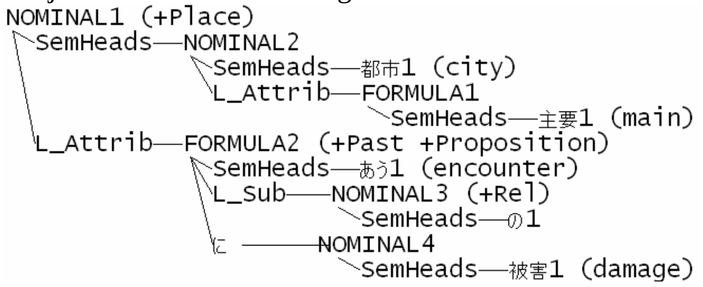


```
NOMINAL1 (+Def +Sing )
 SemHeads—NOMINAL2
            SemHeads—isotope1
            L_Attrib—FORMULA1
                       SemHeads—naturel1
 L_Attrib—FORMULA2 (+Supr +PosSupr)
            SemHeads—lourd1
```

Though the order of modifiers is the opposite from English, the LNS is structurally identical to (8). The LNS abstracts away from the arbitrary conventions of word order, and replaces it with a representation of the logical scope of the modifiers.

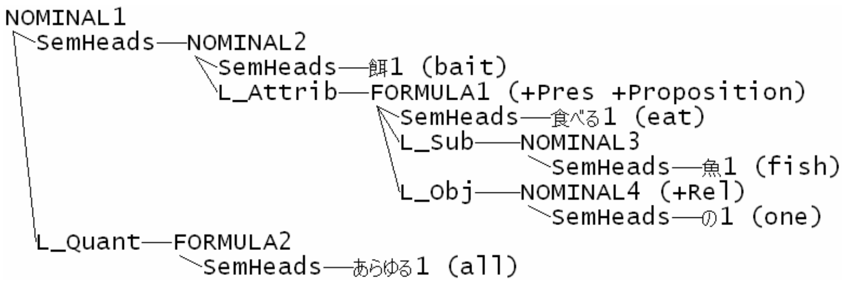
In (8) and (23) the linear order of the modifiers reflects their relative scope. However, in some cases, language-particular ordering conventions are not based on scope but on more superficial grammatical factors such as syntactic category. In these cases as well, the LNS ignores the superficial ordering conventions in favor of logical scope. For example, Japanese relative clauses typically precede other prenominal modifiers, regardless of their relative scope; this is shown in the following examples:

(24) 被害にあった主要な都市
higai -ni a -tta shuyou-na toshi
damage-DAT encounter-PAST major-ADN cities
'major cities that were damaged'



```
NOMINAL1 (+Place)
 SemHeads—NOMINAL2
            SemHeads—都市1 (city)
            L_Attrib—FORMULA1
                       SemHeads—主要1 (main)
 L_Attrib—FORMULA2 (+Past +Proposition)
            SemHeads—あう1 (encounter)
            L_Sub——NOMINAL3 (+Rel)
                     SemHeads—の1
            に ———NOMINAL4
                     SemHeads—被害1 (damage)
```

(25) 魚が食べるあらゆる餌
sakana-ga tabe-ru arayuru esa
fish -NOM eat -PRES all bait
'all bait that fish eat'



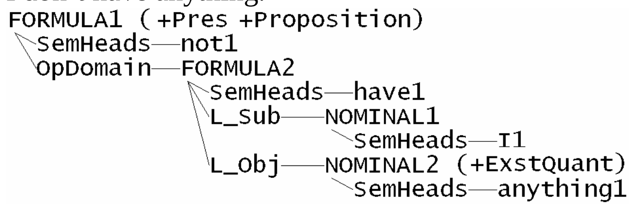
```
NOMINAL1
 SemHeads—NOMINAL2
            SemHeads—餌1 (bait)
            L_Attrib—FORMULA1 (+Pres +Proposition)
                       SemHeads—食べる1 (eat)
                       L_Sub——NOMINAL3
                                SemHeads—魚1 (fish)
                       L_Obj——NOMINAL4 (+Rel)
                                SemHeads—の1 (one)
 L_Quant—FORMULA2
            SemHeads—あらゆる1 (all)
```

In both examples, the relative clause precedes the other modifier; but in (24), it has wider scope than the adjective 主要な *shuyou-na* 'major', while in (25), the quantifier あらゆる *arayuru* 'all' has wider scope. LNS abstracts away from their superficial similarity to show the underlying semantic difference.

3.3 Negation, negative quantifiers and negative polarity items

Negation is another domain which exhibits substantial cross-linguistic variation, especially as it concerns negative quantifiers and adverbs. Sentential negation is represented in LNS as a negative operator taking scope over a sentential constituent, as in (2), above. Consider now (26), containing the negative polarity quantifier *anything*:

(26) I don’t have anything.

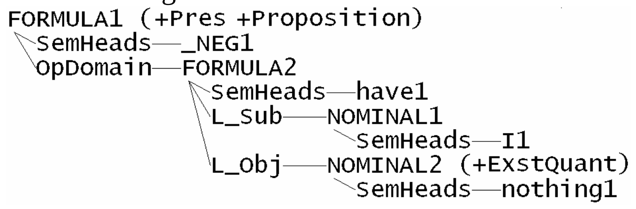


```
FORMULA1 ( +Pres +Proposition)
 \SemHeads—not1
  \OpDomain—FORMULA2
              \SemHeads—have1
              \L_Sub——NOMINAL1
                \         \SemHeads—I1
                 \L_Obj——NOMINAL2 (+ExstQuant)
                           \SemHeads—anything1
```

NOMINAL2 has the feature +*ExstQuant*, which indicates that, despite the lemma of the quantifier, it has the semantic force of an existential; thus (26) is interpreted as ¬∃x[*have*(*I*,x)]; i.e., ‘it is not the case that I have something’.

In both (2) and (26) the negative operator is the English word *not*; in (27), however, there is no word in the surface structure that indicates just negation; instead, negation is part of the meaning of the quantifier *nothing*; in this case, the negative operator is an abstract expression, *_NEG*, which has been factored out of the negative quantifier, leaving behind an existential (i.e., +ExstQuant) quantifier:

(27) I have nothing.

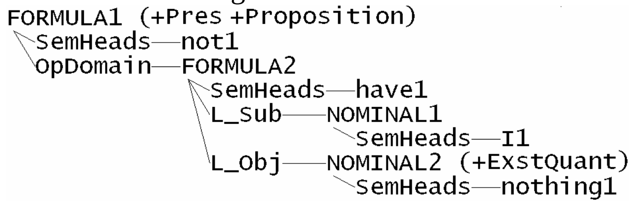


```
FORMULA1 (+Pres +Proposition)
 \SemHeads—_NEG1
  \OpDomain—FORMULA2
              \SemHeads—have1
              \L_Sub——NOMINAL1
                \         \SemHeads—I1
                 \L_Obj——NOMINAL2 (+ExstQuant)
                           \SemHeads—nothing1
```

NOMINAL2 is +ExstQuant in (27), indicating again that it is semantically existential, in spite of having the lemma of a negative word.

The paradigm is completed by (nonstandard) (28), in which there are two negative words, but only one semantic negation, reflected in LNS by a single negative operator:

(28) I don’t have nothing.

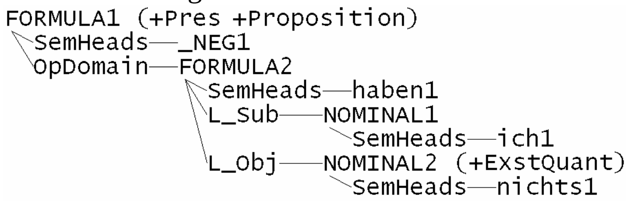


```
FORMULA1 (+Pres +Proposition)
 \SemHeads—not1
  \OpDomain—FORMULA2
              \SemHeads—have1
              \L_Sub——NOMINAL1
                \         \SemHeads—I1
                 \L_Obj——NOMINAL2 (+ExstQuant)
                           \SemHeads—nothing1
```

(26) – (28) have different pairings of negative operator and +ExstQuant quantifier, so their surface difference is recoverable from LNS. But aside from the lemmas of these expressions, the LNSs are identical, reflecting their identical interpretation.

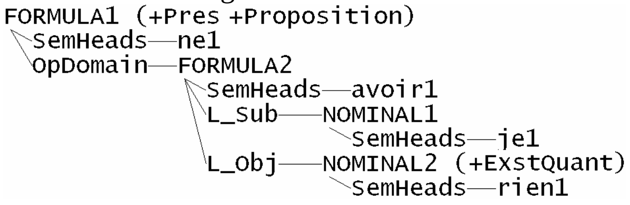
Other languages exhibit a few variations on this paradigm; consider the German, French and Japanese translations of (26), below:

(29) Ich habe nichts.
I have nothing



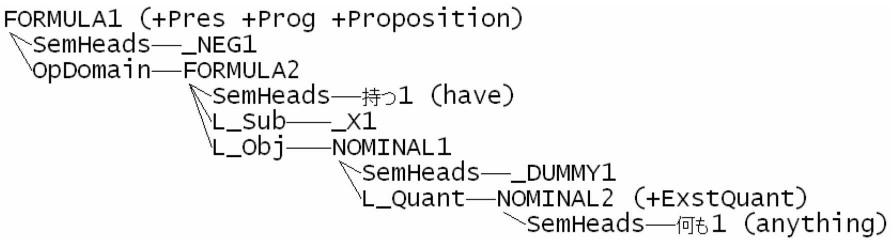
```
FORMULA1 (+Pres +Proposition)
 \SemHeads—_NEG1
  \OpDomain—FORMULA2
             \SemHeads—haben1
             \L_Sub——NOMINAL1
             \          \SemHeads—ich1
              \L_Obj——NOMINAL2 (+ExstQuant)
                        \SemHeads—nichts1
```

(30) Je n' ai rien.
I not-have nothing



```
FORMULA1 (+Pres +Proposition)
 \SemHeads—ne1
  \OpDomain—FORMULA2
             \SemHeads—avoir1
             \L_Sub——NOMINAL1
             \          \SemHeads—je1
              \L_Obj——NOMINAL2 (+ExstQuant)
                        \SemHeads—rien1
```

(31) 何も持っていない。
nani -mo *motte-i* *-nai*
what-NEGPOL have -STATE-NEG
'(I) don't have anything.'

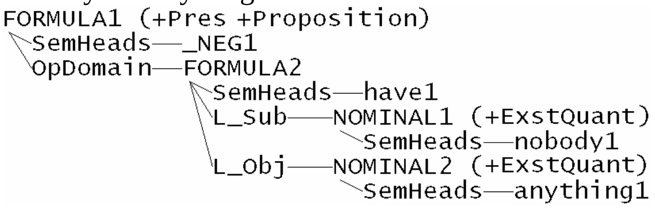


```
FORMULA1 (+Pres +Prog +Proposition)
 \SemHeads—_NEG1
  \OpDomain—FORMULA2
             \SemHeads—持つ1 (have)
             \L_Sub——_X1
              \L_Obj——NOMINAL1
                        \SemHeads—_DUMMY1
                         \L_Quant—NOMINAL2 (+ExstQuant)
                                    \SemHeads—何も1 (anything)
```

German (29) has a surface structure similar to English (27), with a negative quantifier and no separate marker of negation; French (30) is similar on the surface to English (28), with a negative quantifier accompanied by a marker of sentential negation; Japanese (31) has a negative polarity quantifier[5] accompanied by sentential negation (expressed as verbal inflection). In all relevant respects, however, these sentences have the same LNS.

The examples above all involve sentential negation with a single +ExstQuant quantifier; however, multiple +ExstQuant quantifiers can co-occur. Consider the following examples from English and Spanish:

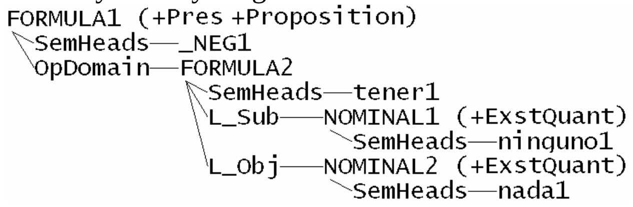
(32) Nobody has anything.



```
FORMULA1 (+Pres +Proposition)
 \SemHeads—_NEG1
  \OpDomain—FORMULA2
             \SemHeads—have1
             \L_Sub——NOMINAL1 (+ExstQuant)
             \          \SemHeads—nobody1
              \L_Obj——NOMINAL2 (+ExstQuant)
                        \SemHeads—anything1
```

[5] Japanese 何も *nani-mo* 'anything' is analyzed as a quantifier modifying an empty head, represented in LNS with the lemma *_DUMMY*. This analysis is motivated by related constructions in Japanese and Chinese, in which the head NP is explicitly mentioned.

(33) Ninguno tiene nada.
nobody has nothing
'Nobody has anything.'



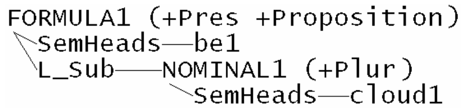
```
FORMULA1 (+Pres +Proposition)
 \SemHeads——_NEG1
  \OpDomain——FORMULA2
               |\SemHeads——tener1
               | \L_Sub——NOMINAL1 (+ExstQuant)
               |          \SemHeads——ninguno1
                \L_Obj——NOMINAL2 (+ExstQuant)
                          \SemHeads——nada1
```

In both examples, *_NEG* has scope over two +ExstQuant quantifiers; in the English (32) one of these comes from a negative quantifier and the other from a negative polarity quantifier, while in the Spanish (33) both are negative quantifiers on the surface. In both cases, there is a single semantic negation, expressed in LNS as a negative operator.[6]

3.4 Constructions with *be*

The English verb *be* has a variety of functions, including copula linking subject and predicate, as in *she is clever;* existential verb, as in *there are clouds*; modal auxiliary, as in *you are to arrive on time*; and passive auxiliary, as in *they were seen*. Each function is treated differently in LNS: the existential verb *be* is treated as a normal intransitive verb:

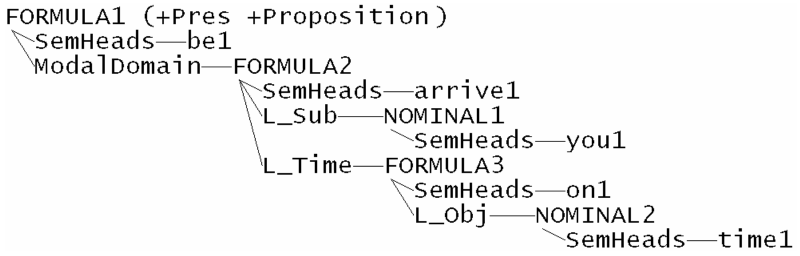
(34) There are clouds.



```
FORMULA1 (+Pres +Proposition)
 \SemHeads——be1
  \L_Sub——NOMINAL1 (+Plur)
           \SemHeads——cloud1
```

The modal auxiliary *be* is treated as a modal operator:

(35) You are to arrive on time.

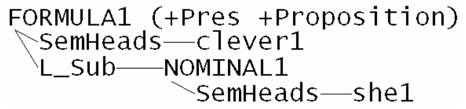


```
FORMULA1 (+Pres +Proposition)
 \SemHeads——be1
  \ModalDomain——FORMULA2
                 |\SemHeads——arrive1
                 | \L_Sub——NOMINAL1
                 |          \SemHeads——you1
                  \L_Time——FORMULA3
                            |\SemHeads——on1
                             \L_Obj——NOMINAL2
                                      \SemHeads——time1
```

Neither the passive auxiliary nor the copula appear as nodes in LNS, however. In the case of passive, all that needs to be recorded is that the construction is passive (+Pass) and whether it has tense; the presence or absence of the auxiliary in English is completely predictable.

The copula is also predictable in English, based on the presence or absence of tense; it carries no lexical information of its own (i.e., no information that is not recorded elsewhere):

(36) She is clever.

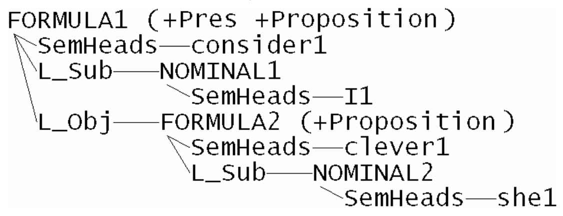


```
FORMULA1 (+Pres +Proposition)
 \SemHeads——clever1
  \L_Sub——NOMINAL1
           \SemHeads——she1
```

[6] True double negation, as in *we can't not do it* ≈ 'we must do it', requires two negative operators.

The existence of the *+Pres* feature on FORMULA1, indicating tense, is sufficient to signal an English generation component that the copula needs to be inserted, since adjectives like *clever* cannot carry tense in English. In the small clause in (37), in contrast, there is no copula in the English surface syntax and in LNS no tense feature on FORMULA2:

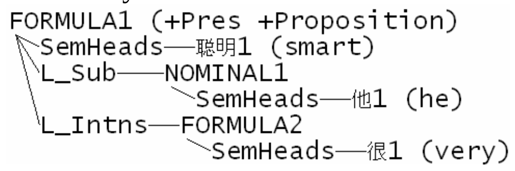
(37) I consider her clever.



```
FORMULA1 (+Pres +Proposition)
 \SemHeads—consider1
  \L_Sub——NOMINAL1
  \         \SemHeads—I1
   \L_Obj——FORMULA2 (+Proposition)
             \SemHeads—clever1
              \L_Sub——NOMINAL2
                        \SemHeads—she1
```

Eliminating the copula serves to normalize English copula constructions and similar constructions in other languages, such as Chinese, which do not use, or do not require, a copula; a Chinese sentence structurally parallel to (37) is given below:[7]

(38) 他很聪明
ta hen congming
he very smart
'He is very smart.'



```
FORMULA1 (+Pres +Proposition)
 \SemHeads—聪明1 (smart)
  \L_Sub——NOMINAL1
  \         \SemHeads—他1 (he)
   \L_Intns—FORMULA2
              \SemHeads—很1 (very)
```

In all relevant respects, the LNSs in (36) and (38) are structurally identical, despite the superficial difference in the presence vs. absence of a copular verb.[8]

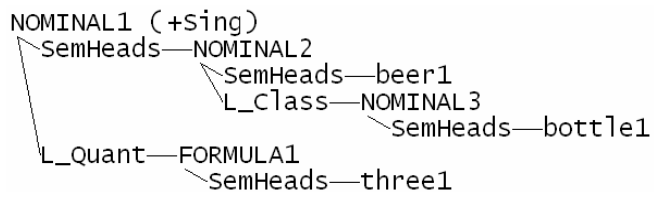
3.5 Classifiers

Although English lacks a system of grammatical classifiers like Chinese and Japanese, there are nevertheless constructions for which a classifier analysis is appropriate. Consider the NP *three bottles of beer*: on the surface, the head of this NP is *bottle*, from which, for example, the phrase inherits its number. In many ways *beer* is the semantic head, however; for instance, *we drank three bottles of beer* entails that we drank beer, not that we drank bottles. In LNS, *beer* is treated as the SemHeads, and *bottle* is assigned to the *L_Class* (logical classifier) attribute:

[7] The +Pres feature in this example is assigned as a default; a more accurate representation of the unmarked tense in Chinese would indicate that no time reference is grammatically determined. See Campbell *et al.* (2002) for a proposal along these lines.

[8] For languages with multiple copulas, such as Spanish *ser* and *estar*, the lemma of the copula can be stored away in a separate non-tree attribute, to facilitate recovery of this information. It is not clear to what extent this is necessary, however: Melero *et al.* (2002) report high accuracy in predicting the lemma of the Spanish copula based on information taken from LNS.

(39) three bottles of beer



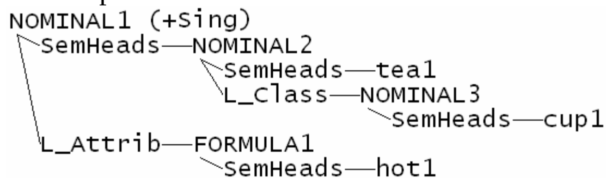
```
NOMINAL1 (+Sing)
 \SemHeads—NOMINAL2
 |           \SemHeads—beer1
 |            \L_Class—NOMINAL3
 |                       \SemHeads—bottle1
  \L_Quant—FORMULA1
            \SemHeads—three1
```

This analysis serves to normalize the analysis English *three bottles of beer* and Chinese NPs such as 三瓶啤酒 *san ping pijiu* 'three bottles of beer', in which 啤酒 *pijiu* 'beer' is both the surface-syntactic head and the SemHeads.

The analysis illustrated in (39) is also motivated on language-internal grounds: As noted, *beer* behaves as the semantic head with respect to semantic selection; also, attributive adjectives that modify the classifier + noun structure are sometimes interpreted as modifying the semantic head noun, as in *a hot cup of tea* or *a stale box of crackers*. These facts together are symptoms of the fact that the NP in (39) refers to beer, and not to bottles; this semantic analysis is transparently reflected in (39).

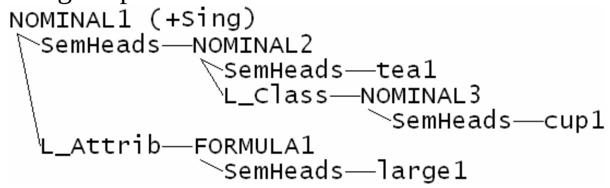
Note, incidentally, that attributive modifiers of classifier + noun constructions do not always appear to modify the noun, but sometimes appear to modify the classifier, as in *a large cup of tea*. These constructions do not differ syntactically, however, and in keeping with the view of LNS as a syntactic representation (see Section 4), they are not distinguished in LNS:

(40) a hot cup of tea



```
NOMINAL1 (+Sing)
 \SemHeads—NOMINAL2
 |           \SemHeads—tea1
 |            \L_Class—NOMINAL3
 |                       \SemHeads—cup1
  \L_Attrib—FORMULA1
             \SemHeads—hot1
```

(41) a large cup of tea



```
NOMINAL1 (+Sing)
 \SemHeads—NOMINAL2
 |           \SemHeads—tea1
 |            \L_Class—NOMINAL3
 |                       \SemHeads—cup1
  \L_Attrib—FORMULA1
             \SemHeads—large1
```

Structurally the adjective modifies *cup of tea* in both cases; the fact that the cup must be large for a cup of tea to be large, and that the tea must be hot for a cup of tea to be hot, are inferences that derive from lexical semantic and/or pragmatic considerations, but that are not structurally represented.

3.6 Future development

LNS, as it currently stands, is reasonably well-developed with respect to the representation of grammatical relations and the functions of operators such as negation; other aspects of the representation are under active development. Campbell *et al*. (2002) propose a system for representing tense in LNS, though their system has not yet been fully adopted into NLPWin, and is not used in the present paper. Some discourse-related aspects of syntactic structure, such as topic/comment structure, focus and presupposition, are partially encoded in LNS, but currently not well-utilized. Other areas also under development include the representation of voice, comparatives, ellipsis, intersentential anaphora and the representation of ambiguity.

4 LNS as a syntactic representation

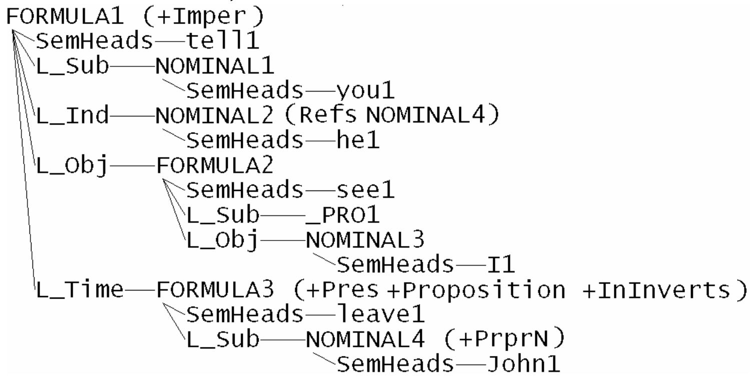
The focus of the previous section was on LNS being sufficiently abstract to provide normalized representations of superficially different structures in different languages or even within the same language. However, LNS must also be concrete enough to preserve possibly meaningful grammatical distinctions in individual languages, so that such distinctions can be derived directly from LNS without consulting the surface syntactic structure. This requirement was expressed as the second design criterion of LNS in Section 1: LNS must preserve potentially meaningful surface distinctions so that they are recoverable from LNS.

This relative shallowness of representation at LNS is beneficial not only in avoiding the problem of brittleness in performing deep semantic analysis for a realistically broad range of input (see Section 1 regarding the interpretation of *black cat* vs. *legal problem*), but it is also a requirement if LNS is to be used as the basis for transfer in transfer-based MT:[9] to be a useful basis for generation, it is necessary to preserve information about meaningful surface distinctions. Ideally, the best way to represent meaningful grammatical distinctions in LNS is for the representation to be semantically transparent: thus for example order of modifiers is represented as scope, as discussed in Section 3. However, in many cases, we simply do not know how to represent the semantic difference between two different forms, even though we are sure there is such a difference. In such a case, LNS needs to be annotated in some way to indicate the difference. In this section, we discuss three such cases, where LNS remains faithful to the surface input, while still providing abstract representations as above.

4.1 Pronominalization

Consider the following two sentences:

(42) Before John leaves, tell him to see me.



```
FORMULA1 (+Imper)
  SemHeads—tell1
  L_Sub——NOMINAL1
             SemHeads—you1
  L_Ind——NOMINAL2 (Refs NOMINAL4)
             SemHeads—he1
  L_Obj——FORMULA2
             SemHeads—see1
             L_Sub——_PRO1
             L_Obj——NOMINAL3
                        SemHeads—I1
  L_Time—FORMULA3 (+Pres +Proposition +InInverts)
             SemHeads—leave1
             L_Sub——NOMINAL4 (+PrprN)
                        SemHeads—John1
```

[9] In fact, MSR-MT, a transfer-based MT system, has been the major application that has driven the development of LNS. For details about MSR-MT, see Richardson et al. (2001).

(43) Before he leaves, tell John to see me.

```
FORMULA1 (+Imper)
 \SemHeads—tell1
 \L_Sub——NOMINAL1
 |         \SemHeads—you1
 \L_Ind——NOMINAL2 (+PrprN)
 |         \SemHeads—John1
 \L_Obj——FORMULA2
 |         \SemHeads—see1
 |         \L_Sub——_PRO1
 |         \L_Obj——NOMINAL3
 |                   \SemHeads—I1
 \L_Time—FORMULA3 (+Pres +Proposition +InInverts)
           \SemHeads—leave1
           \L_Sub——NOMINAL4 (Refs NOMINAL2)
                     \SemHeads—he1
```

These two examples illustrate one way the LNS records surface information. In (42), the L_Ind of FORMULA1 is a pronoun and the coreferential L_Sub of FORMULA3 is a full NP, while in (43) the reverse is true. In both cases, the non-tree attribute *Refs* of the pronoun points to *John*, indicating the (possible) coreference between the two, but LNS preserves the information as to which is pronominal.[10]

4.2 Words and word senses

LNS is also relatively close to surface syntax in that word senses are not disambiguated, unless such disambiguation is grammatically motivated; as a result, the leaf nodes of an LNS tree are the lexemes of words in a particular language (and labeled with that lexeme's lemma). The main advantage of this approach is that LNS is closely bound with surface syntax; in particular, surface information about which words were used in a given sentence is directly recoverable from the LNS for that sentence.

In some cases, two senses are distinguished grammatically; in such cases, what is involved are not distinct senses but distinct lexemes. Currently LNS supports two methods of distinguishing lexemes with the same lemma. To handle cases that are distinguished by part of speech, LNS makes use of the non-tree *Cat* attribute (not displayed in the LNS diagrams above) to record part of speech.[11] Consider the following examples, in which the Cat of each lexical item is displayed:

(44) She likes me.

```
FORMULA1 (+Pres +Proposition)
 \SemHeads—like1 ({Verb})
 \L_Sub——NOMINAL1
 |         \SemHeads—she1 ({Pron})
 \L_Obj——NOMINAL2
           \SemHeads—I1 ({Pron})
```

(45) She is like me.

```
FORMULA1 (+Pres +Proposition)
 \SemHeads—like1 ({Adj})
 \L_Sub——NOMINAL1
 |         \SemHeads—she1 ({Pron})
 \L_Obj——NOMINAL2
           \SemHeads—I1 ({Pron})
```

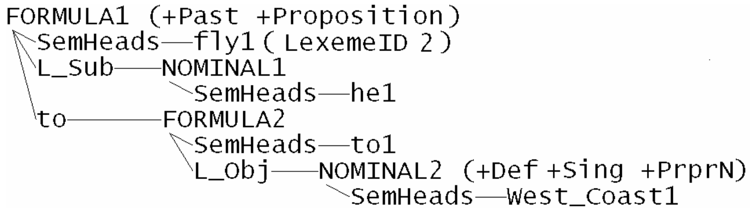
[10] Another difference between these two examples is that FORMULA2, corresponding to the subordinate clause, has the feature +InInverts in both examples, indicating that it is preposed.

[11] Cat was developed independently in NLPWin for use with other semantic representations that predate the development of LNS.

These LNSs differ only in the Cat of *like1*.

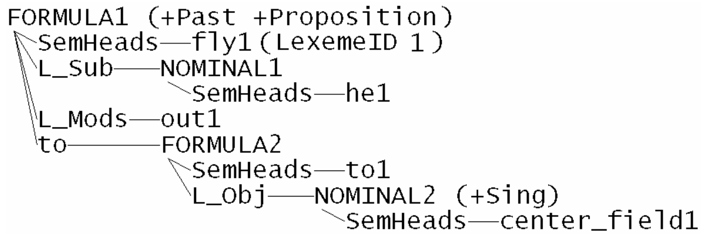
The non-tree attribute *LexemeID* (also not displayed above) distinguishes lexemes that have the same part of speech, by assigning an arbitrary index to each; this index can be thought of as a pointer to a particular dictionary entry, though a dictionary is not strictly necessary for LexemeID to serve the purpose of disambiguation.[12] Consider the following examples:

(46) He flew to the West Coast.



```
FORMULA1 (+Past +Proposition)
 \SemHeads—fly1 ( LexemeID 2 )
 \L_Sub——NOMINAL1
 \         \SemHeads—he1
  \to———FORMULA2
             \SemHeads—to1
              \L_Obj——NOMINAL2 (+Def +Sing +PrprN)
                        \SemHeads—West_Coast1
```

(47) He flied out to center field.



```
FORMULA1 (+Past +Proposition)
 \SemHeads—fly1 (LexemeID 1 )
 \L_Sub——NOMINAL1
 \         \SemHeads—he1
 \L_Mods—out1
  \to———FORMULA2
             \SemHeads—to1
              \L_Obj——NOMINAL2 (+Sing)
                        \SemHeads—center_field1
```

In both case the lemma of the head verb is *fly*; in (46), this node has a LexemeID of 2, which is a pointer to the irregular verb *fly,* whose preterite is *flew*; in (47), the LexemeID of *fly* is 1, pointing to the regular verb *fly* (a baseball term) whose preterite is *flied*. A system that did not make use of an English dictionary could still make use of the LexemeIDs in these examples, however, since the distinct LexemeIDs in (46) and (47) serve, by themselves, to indicate that the verbs are distinct lexemes. In addition to part-of-spech and inflectional classes, other morphological features such as gender can also contribute to grammatically motivated word sense disambiguation; within NLPWin, these are all represented using LexemeIDs.

Beyond grammatically motivated cases like these, however, there is no word sense disambiguation, nor representation of senses as such (Dolan *et al.*, 2000). This is by design: recall from Section 1 that LNS is intermediate between surface syntax and more abstract semantic analyses (see also Section 5, below); therefore, if further sense-disambiguation is possible, and is required by a given application, it can be done in a more abstract semantic representation derived from LNS, rather than in LNS itself. However, for most applications, it does not appear to be necessary; for example, in the MSR-MT system, transfer mappings are learned automatically on the basis of aligned representations, so sense-dependent translations are learned from context. For example, consider the English verb *fail*, which might be thought of as having two (or more) senses: In a sentence like *he failed*, it means not to succeed at some task; in a sentence like *the network failed*, it means to cease operating. These sentences would be translated into Japanese differently: 彼は失敗した *kare-wa <u>shippai-sita</u>* 'he failed' vs. ネットワークに障害が発生した *nettowaaku-ni <u>shougai-ga hassei-shita</u>* 'the network failed' (lit. 'difficulty occurred in the network'). Although word senses are not represented in LNS, the MSR-MT system learns from a bilingual, aligned corpus that in some contexts *fail* translates as 失敗 *shippai* 'fail' and in other

[12] Lexemes as described in the text, and the associated attribute LexemeID, were developed for the NLPWin system by Joseph Pentheroudakis independently of LNS; LexemeID has simply been adopted into LNS unchanged.

contexts translates as 障害が発生 *shougai-ga hassei* ‘difficulty occurs’. Thus context is a substitute for word sense – a much less brittle approach that does not depend on a theory of senses.

4.3 Punctuation

Sentence-final punctuation is often meaningful; the difference between period/full stop and question mark in English is an obvious example. In this case, the difference can be represented as the different clause type features (Section 2.5). In real text, however, there are often further variations in sentence-final punctuation for which there may be no obvious semantic representation; to handle such cases, LNS representations can be annotated with a non-tree attribute *SentPunc*, which simply records the form of the final (or initial) punctuation. Note that the value of SentPunc is not a node in the tree, but simply a list of the punctuation marks. This is illustrated by the following examples:

(48) I left.

```
FORMULA1 (+Past +Proposition SentPunc (.))
 \SemHeads—leave1
  \L_Sub——NOMINAL1 (+Sing)
                 \SemHeads—I1
```

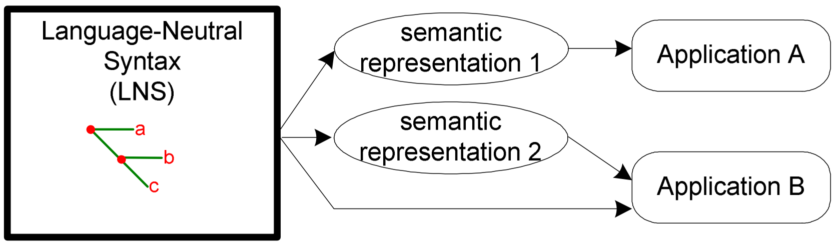
(49) I left...

```
FORMULA1 (+Past +Proposition SentPunc (...))
 \SemHeads—leave1
  \L_Sub——NOMINAL1 (+Sing)
                 \SemHeads—I1
```

(48) and (49) differ only in the final punctuation; it is not clear how to represent this difference in a semantically meaningful way. The LNSs for these sentences differ only in the value of the attribute SentPunc, which records the punctuation directly.

5 LNS and semantic representation

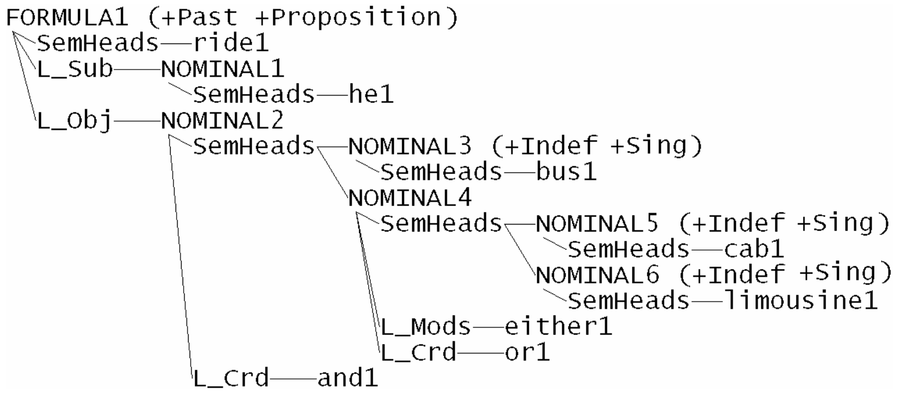
LNS can be the input to a semantic representation, as outlined in Section 1. Indeed, there may be multiple semantic representations, which can be derived from LNS, each required by different applications perhaps, expressing different kinds of semantic properties (Campbell and Suzuki, 2002). The following diagram shows this schematically: different semantic representations are derivable from LNS, and various applications may make use of specific semantic representations, and/or LNS itself:


Language-Neutral Syntax (LNS)
a
b
c
semantic representation 1
semantic representation 2
Application A
Application B

The basic principles of LNS design require that the LNS of a given sentence contain as much information about its surface syntax as is needed to derive such representations from LNS without additional surface-syntactic information. In principle, therefore, any semantic representation that could be derived from surface syntax can be derived from LNS by a language-independent function.

One example of a semantic representation that is currently in use in the NLPWin system is Predicate-Argument Structure (PAS), a graph showing the lexical dependencies inherent in LNS in a strictly local fashion.[13] Consider for example the sentence *he rode a bus and either a cab or a limousine*, which has the LNS shown in (50):

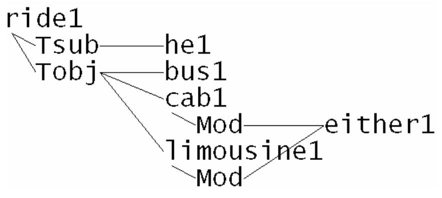
(50) He rode a bus and either a cab or a limousine.



```
FORMULA1 (+Past +Proposition)
 SemHeads—ride1
 L_Sub——NOMINAL1
          SemHeads—he1
 L_Obj——NOMINAL2
          SemHeads—NOMINAL3 (+Indef +Sing)
                    SemHeads—bus1
                   NOMINAL4
                    SemHeads—NOMINAL5 (+Indef +Sing)
                              SemHeads—cab1
                             NOMINAL6 (+Indef +Sing)
                              SemHeads—limousine1
                    L_Mods—either1
                    L_Crd——or1
          L_Crd——and1
```

The relation between *ride* and the various nouns in the coordinate NP is indirect; and in general the path between say a predicate and the various conjoined nouns in that predicate's argument is arbitrarily long in LNS. A given application may need to make use of such relations, however, e.g. in determining that *bus*, *cab* and *limousine* are all things that one commonly rides. PAS provides just such a representation; the PAS for (50) is shown below:

(51) He rode a bus and either a cab or a limousine.

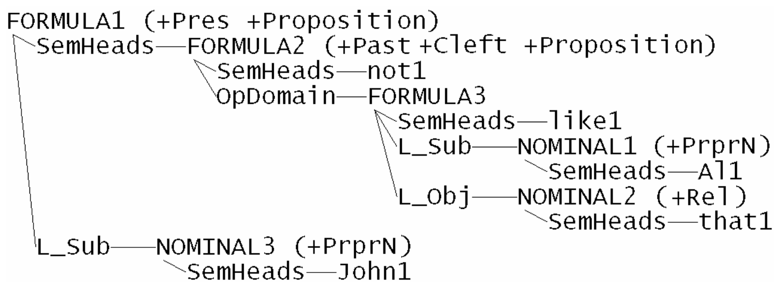


```
ride1
 Tsub———he1
 Tobj———bus1
        cab1
         Mod———either1
        limousine1
         Mod
```

In this representation, all three nouns are the value of the PAS-only attribute *Tobj* of *ride1*, indicating that they are typical objects of *ride*. No matter how complex the coordinate structure in LNS, in PAS, which represents only the lexical dependencies, the structure is flattened.

A representation in predicate calculus could also be derived from an LNS (with the caveats sketched in Section 1). Although this has not been implemented for NLPWin, it is straightforward, since LNS preserves the relevant parts of surface syntax. Consider for example the LNSs for *it's John that Al didn't like* and *it's not John that Al liked*:

(52) It's John that Al didn't like.



```
FORMULA1 (+Pres +Proposition)
 SemHeads—FORMULA2 (+Past +Cleft +Proposition)
           SemHeads—not1
           OpDomain—FORMULA3
                     SemHeads—like1
                     L_Sub——NOMINAL1 (+PrprN)
                             SemHeads—Al1
                     L_Obj——NOMINAL2 (+Rel)
                             SemHeads—that1
 L_Sub——NOMINAL3 (+PrprN)
         SemHeads—John1
```

[13] PAS corresponds to *Logical Form* as that term is used by Heidorn (2000).

(53) It’s not John that Al liked.

```
FORMULA1 (+Pres +Proposition)
 \SemHeads—not1
  \OpDomain—FORMULA2
             |\SemHeads—FORMULA3 (+Past +Cleft +Proposition)
             |           |\SemHeads—like1
             |           |\L_Sub——NOMINAL1 (+PrprN)
             |           |         \SemHeads—Al1
             |            \L_Obj——NOMINAL2 (+Rel)
             |                     \SemHeads—that1
              \L_Sub——NOMINAL3 (+PrprN)
                       \SemHeads—John1
```

These LNSs can be straightforwardly translated into predicate calculus; for illustration, we follow the Intensional Predicate Calculus notation used by Chierchia and McConnell-Ginet (1990), treating names as constant terms, verbs as predicates of the appropriate adicity, and tenses as intensional operators at the sentence level; the representations derived are shown in (54) and (55), respectively:[14]

(54)	$[\lambda x[\mathbf{P}\neg \mathit{like}(\mathit{Al},x)](\mathit{John})]$	‘It’s John that Al didn’t like.’
(55)	$\neg[\lambda x[\mathbf{P}\mathit{like}(\mathit{Al},x)](\mathit{John})]$	‘It’s not John that Al liked.’

Moreover, as noted in Section 2, presuppositions of cleft sentences can be derived by existential closure over the LNS constituent that is the sister of L_Foc; i.e., $\exists x[\mathbf{P}\neg \mathit{like}(\mathit{Al},x)]$ (i.e., there is someone or something that Al didn’t like) is a presupposition of (52); and $\exists x[\mathbf{P}\mathit{like}(\mathit{Al},x)]$ (i.e., Al liked someone or something) is a presupposition of (53).[15]

PAS and other specialized semantic representations are not part of the LNS *per se*, and need not be explicitly specified in LNS. Instead, as noted above, they are derived from LNS by language-independent functions.

6 Comparison to other frameworks

Many frameworks that claim to be language-neutral representations for multi-lingual applications are based on a (possibly underspecified) logical representation. Representational frameworks of this kind include QLF (Alshawi *et al.*, 1991; Alshawi and Crouch, 1992), UDRS (Reyle, 1993), Language for Underspecified Discourse representations (Bos, 1995), Minimal Recursion Semantics (Copestake *et al.*, 1995, 1999) and the Logical Form language of Allen (1995). The distinguishing features of such representations include the use of word-senses as logical predicates, and the explicit logical representation of relations among constituents, such as the relation of an attributive adjective to a head noun. LNS differs from them in both respects: the leaf nodes of an LNS tree are lexemes, not word-senses (Section 4.2), and there is no attempt to logically characterize all modification relations: *black cat* and *legal problem* are assigned the same LNS structure, despite their semantic difference (Section 1).

Among such frameworks, LNS is closest in conception to QLF, insofar as it purports to be a representation that is intermediate between surface syntax and deeper semantic

[14] Although tense is not currently represented as a separate node in LNS, we assume for the purposes of illustration that a tense feature on a constituent C takes wider scope than any operator inside C; thus in the example in the text the past tense operator **P** is assigned wider scope than the negative operator.

[15] Note that, since quantifiers like *every* and *some* are not assigned scope in LNS, a fully specified logical representation of sentences containing them would require additional processing beyond LNS. A more reasonable logical representation to derive from LNS would therefore be an underspecified representation, such as QLF (Alshawi *et al.*, 1991) or UDRS (Reyle, 1993).

representations. However, QLF does not attempt to normalize meaningless cross-linguistic differences in word order; for example, English and Swedish differ in the placement of sentential negation relative to the finite verb:

(56) John doesn’t like Mary.

(57) John tycker inte om Mary.
J. thinks not about M.

According to Alshawi *et al*. (1991), present tense is assigned wider scope than negation in (56), while the reverse is true in (57); in both cases, the relative scope of tense and negation in QLF reflects the relative order of finite verb and negation on the surface. Yet the same paper recognizes that the difference is meaningless, and introduces a special transfer rule to reorder tense and negation in translation between the two languages. Our view, in contrast, is that the various orderings of tense and negation are language-particular grammatical devices, not represented in LNS. As such, an MT system using LNS would have no need for a special transfer rule.

As an abstract syntactic representation, LNS is reminiscent of deep syntactic representations such as f-structure in Lexical Functional Grammar (Bresnan 1982, 2001) and DSyntS (Lavoie and Rambow 1997) based on Dependency Syntax (Mel'čuk 1988). Both these representations try to encode syntactic structure using a language-neutral formal vocabulary, and their benefit to applications such as MT has also been explored (e.g., Dorna *et al*., 1998 using f-structure, and Han *et al*., 2001 using DSyntS). DSyntS is similar to LNS, and different from many of the representations mentioned above, in that it is an unordered tree with labeled arcs and its leaf nodes are lexemes from the analyzed language. It differs from LNS, however, in two respects: (1) there are no grammatical variables such as relative pronouns and *_PRO* as in LNS (Section 2.4), and (more crucially) (2) there are no non-terminal nodes, hence no attempt to represent the scope of logical operators and modifiers, even where this is syntactically determined. In fact, DSyntS is most similar to the PAS representation of Section 5.

F-structure of Lexical Functional Grammar is another framework for representing syntactic structure, though the similarity between f-structure and LNS is almost limited to the fact that they both normalize surface word order and certain functional information. Similarly to DSyntS, f-structure contains no non-terminal nodes, and does not preserve the scope of logical operators and modifiers; unlike DSyntS and LNS, the grammatical relations in f-structure are *surface* grammatical relations, not normalized for operations such as passivization. There is also a fundamental difference between f-structure and LNS regarding the level of abstraction: LFG is strongly lexicalist in that the values of the PRED attribute in an f-structure are semantic objects, and f-structure is subject to lexically-based (i.e., word sense-based) well-formedness conditions such as Completeness and Coherence[16]. In contrast, LNS is based on words and not on word senses (see Section 4.2), in order to enable robust mapping from surface syntax on a realistically broad range of input.

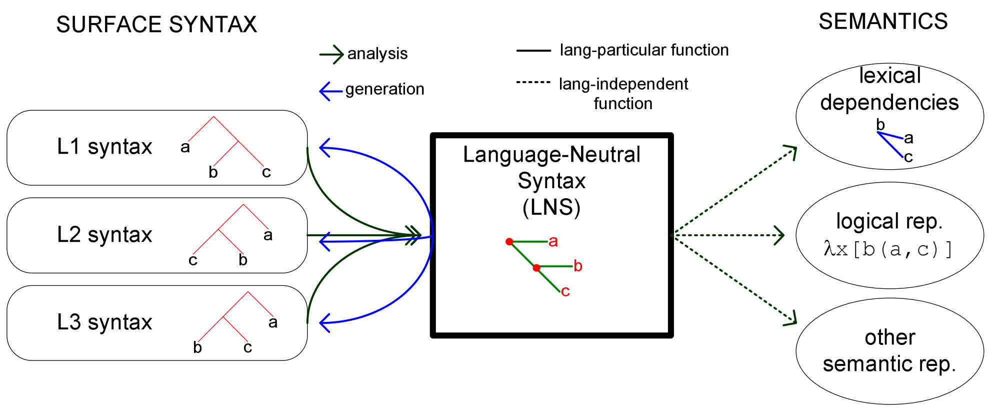
7 Conclusion

LNS is a system of representation that is intermediate between language-particular, surface-based syntax on the one hand, and abstract, fully-articulated (though perhaps underspecified) semantic representations on the other hand. Expressions, either in the same language or in different

[16] The Completeness condition states that every function designated by a PRED attribute must be present in the f-structure, while the Coherence condition requires that every argument function in an f-structure be designated by a PRED (Bresnan, 2001: 63). These conditions together ensure that there are no missing or extra arguments in an f-structure given the word sense of the predicate (PRED).

languages, that have the same essential structure are represented the same in LNS (up to the lexicon), regardless of either their superficial syntactic differences or their deep semantic differences. LNS is abstract enough to be language-neutral, yet shallow enough to preserve essential elements of surface syntax and lexical choices, allowing the derivation of more specialized semantic representations as necessary.

The following diagram illustrates the relation between LNS and other representations that may be used: Language-particular syntactic representations are normalized to LNS in analysis, or generated from LNS; various semantic representations can be derived from LNS if needed for particular applications:



A representation such as LNS, we claim, serves more effectively than either surface syntax or semantics as a common representation schema for a variety of applications, especially multilingual ones.

Acknowledgments

We gratefully acknowledge the help and advice of Mike Calcagno, Cassandre Creswell, and our colleagues in the NLP group at Microsoft Research, especially Bob Moore and Lucy Vanderwende. All errors and inconsistencies are of course our own.

References

Allen, J. 1995. *Natural language understanding*. Redwood City, CA: Benjamin/Cummings.

Alshawi, H., D. Carter, M. Rayner and B. Gambäck. 1991. Translation by Quasi Logical Form transfer. In *Proceedings of ACL 29*: 161-168.

Alshawi, H., and R. Crouch. 1992. Monotonic semantic interpretation. In *Proceedings of ACL 30*.

Bos, J. 1995. Predicate logic unplugged. In *Proceedings of the 10th Amsterdam colloquium*, University of Amsterdam.

Bouillon, P. and E. Viegas. 1999. The description of adjectives for Natural Language Processing: Theoretical and applied perspectives. In P. Bouillon and E. Viegas (eds.), *Atelier Thématique TALN 1999*.

Bresnan, J. (ed.). 1982. *The mental representation of grammatical relations*. Cambridge: MIT Press.

Bresnan, J. 2001. *Lexical-Functional Syntax*. Malden, MA and Oxford: Blackwell.

Campbell, R. 2002. Computation of modifier scope in NP by a language-neutral method. In *Proceedings of COLING 2002*.

Campbell, R., T. Aikawa, Z. Jiang, C. Lozano, M. Melero and A. Wu. 2002. A language-neutral representation of temporal information. Workshop on annotation standards for temporal information in natural language, LREC 2002.

Campbell, R. and H. Suzuki. 2002. Language-neutral representation of syntactic structure. In R. Malaka, R. Porzel and M. Stube (eds.), *Proceedings of SCANALU 2002*.

Chierchia, G. and S. McConnell-Ginet. 1990. *Meaning and grammar: An introducton to semantics*. Cambridge: MIT Press.

Copestake, A., D. Flickinger, R. Malouf, S. Riehemann and I. Sag. 1995. Translation using Minimal Recursion Semantics. In *Proceedings of TMI-95*.

Copestake, A., D. Flickinger, I. Sag and C. Pollard. 1999. *Miminal Recursion Semantics: An introduction*. Ms., Stanford University.

Dolan, B., L Vanderwende and S. Richardson. 2000. Polysemy in a broad-coverage natural language processing system. In Y. Ravin and C. Leacock (eds.), *Polysemy: Theoretical and computational approaches*. Oxford: Oxford University Press. pp.178-204.

Dorna, M., A. Frank, J. van Genabith and M. Emele. 1998. Syntactic and Semantic Transfer with F-Structures. In *Proceedings of COLING-ACL '98*: 341-347.

Han, C-H., B. Lavoie, M. Palmer, O. Rambow, R. Kittredge, T. Korelsky, N.Kim and M. Kim. 2001. Handling structural divergences and recovering dropped arguments in a Korean/English machine translation system. In *Proceedings of AMTA*.

Heidorn, G. 2000. Intelligent writing assistance. In R. Dale, H. Moisl and H. Somers (eds.), *A handbook of natural language processing: Techniques and applications for the processing of language as text*, New York: Marcel Dekker.

Keenan, E.L. and L.M. Faltz. 1985. *Boolean semantics for natural language*. Dordrecht: D. Reidel.

Lavoie, B. and O. Rambow. 1997. A fast and portable realizer for text generating systems. In *Proceedings of ANLP'97*.

Melero, M., T. Aikawa and L. Schwartz. 2002. Combining machine learning and rule-based approaches in Spanish and Japanese sentence realization. Second International Natural Language Generation Conference, New York.

Mel'čuk, I. 1988. *Dependency Syntax: Theory and practice*. New York: State University of New York Press.

Reyle, U. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics* 10: 123-179.

Richardson S., W. Dolan, A. Menezes and J. Pinkham. 2001. Achieving commercial-quality translation with example-based methods. In *Proceedings of the VIIIth MT summit*, Santiago de Compostela, Spain. 293-298.

Appendix

Table I: Basic tree attributes: if x == attr(y), then y is x's parent

Attribute	**Usage**	**Examples**
L_Sub	"logical subject": agent, actor, cause or other underlying subject relation; not e.g. subject of passive, raising, or unaccusative predicate; also used for subject of predication	**She** took it; **John** ran; It was done by **me**; **you** are tall.
L_Ind	"logical indirect object": goal, recipient, benefactive	I gave it to **her**; **I** was given a book
L_Obj	"logical (direct) object": theme, patient, including e.g. subject of unaccusative; also object of preposition	She took **it**; **The window** broke; **He** was seen by everyone

L_Pred	"logical predicate": secondary predicate, e.g. resultative or depictative	We painted the barn **red**; I saw them **naked**
L_Loc	location	I saw him **there**
L_Time	time when	He left **before I did**; He left **at noon**
L_Dur	duration	I slept for **six hours**
L_Caus	cause or reason	I slept **because I was tired**; She left **because of me**
L_Poss	possessor	**my** book; some friends of **his**
L_Quant	quantifier/determiner	**three** books; **every** woman; **all** of them; the **other** people
L_Mods	otherwise unresolved modifier	I left **quickly**
L_Crd	conjunction in coordinate structure	John **and** Mary
L_Interlocs	interlocutor(s), addressee(s)	**John**, come here!
L_Appostn	appositive	John, **my friend**, left
L_Purp	purpose clause	I left **to go home**; His wife drove **so that he could sleep**; I bought it **in order to please you**
L_Intns	intensifier	He was **very** angry.
L_Attrib	attributive modifier (adjective, relative clause, or similar function)	the **green** house; the woman **that I met**.
L_Means	means by which	He covered up by **humming**.
L_Class	classifier; often this is the grammatical head but not the logical head	a **box** of crackers
OpDomain	scope domain of a sentential operator	**He did** not **leave**
ModalDomain	scope domain of a modal verb/particle	**I** must **leave**.
SemHeads	logical function: head or sentential operator	He did **not** leave; my good **friend**; He **left**.
Ptcl	particle forming a phrasal verb	He gave **up** his rights

Table II: Basic non-tree attributes

Attribute	**Type of value**	**Usage**	**Attribute of**
Cntrlr	single node	controller or binder of dependent element	dependent item
L_Top	list of nodes	logical topic	clause
L_Foc	list of nodes	focus, e.g. of pseudo(cleft)	clause
PrpObj	single node	object of pre/postposition (often also L_Obj; see Table I)	node headed by pre/postposition
Nodename	string	unique name/label of an LNS node; the value of Nodename is the value of Pred (for terminal nodes) or Nodetype (for nonterminal nodes) followed by an integer unique among all the nodes with that Pred or Nodetype.	all nodes

Nodetype	string	FORMULA or NOMINAL or null; all and only non-terminal nodes have a Nodetype	all non-terminal nodes
Pred	string	for terminal nodes, Pred is the lemma	terminal nodes
MaxProj	single node	maximal projection; every node, whether terminal or nonterminal, should have one	all nodes
Refs	list of nodes	list of possible antecedents for pronominals and similar nodes	anaphoric expression
Cat	string	part of speech	terminal nodes
SentPunc	list of strings	sentence-level punctuation	root sentence

Table III: Basic LNS features

Feature name	**Usage**	**Examples**
Proposition	[+Proposition] identifies a node to be interpreted as having a truth value; declarative statement, whether direct or indirect	**I left**; I think **he left**; I believe **him to have left**; I consider **him smart**; NOT E.G. I saw **him leave**; **the city's destruction** amazed me
YNQ	identifies a node that denotes a yes/no question, direct or indirect	**Did he leave?**; I wonder **whether he left**
WhQ	identifies a node that denotes a wh-question, direct or indirect; marks the scope of a wh-phrase in such a question	**Who left?**; I wonder **who left**
Imper	imperative	**Leave now!**
Def	definite	The **plumber** is here
Sing	singular	dog; mouse
Plur	plural	dogs; mice
Pass	passive	she **was seen**
ExstQuant	indicates that a quantifier or conjunction has existential force, regardless of the lexical value; e.g. in negative sentence with negative or negative-polarity quantifiers; not used with existential quantifiers that regularly have existential force (e.g. *some*); see Section 3.3.	We (don't) need **no** badges; We don't need **any** badges
Reflex	reflexive pronoun	He admired **himself**
ReflexSens	reflexive sense of a verb distinct from non-reflexive senses	He **acquitted himself** well
Cleft	kernel (presupposed part) of a (pseudo)cleft sentence	It was her **that I met**; **who I really want to meet** is John
Comp	comparative adjective or adverb	
Supr	superlative adjective or adverb	
NegComp	negative comparative	**less well**
NegSupr	negative superlative	**least well**
PosComp	positive comparative	**better**
PosSupr	positive superlative	**best**
AsComp	equative comparative	**as good as**