A Sequential Approach to Lexical Sign Description

Michael Filhol, Annelies Braffort
LIMSI/CNRS
BP 133, F-91403 Orsay cedex
E-mail: michael.filhol@limsi.fr, annelies.braffort@limsi.fr

Abstract

Sign description systems able precisely to detail how a lexical unit of a sign language is performed are not that numerous. Plus, in the prospect of implementing such a description model for automatic sign generation by virtual characters, visual notation systems such as SignWriting, however accurate they are, cannot be used. The Hamburg Notation System (HamNoSys) (Hanke, 1989) together with its more computer-friendly super-set SiGML (Signing Gesture Markup Language) is about as advanced a model we could find, and yet some problems still have to be tackled in order to obtain an appropriate sign description system. Indeed, based on Stokoe-type parameters, it assumes every sign can/must be described with the same fixed set of parameters, each of which would be given a discrete value. However, we argue that not all signs require all parameters, and that not all the parameters that are needed can be given at the same time in the same way. This work underlines three problems we see with Stokoe-like descriptions, and suggests a new approach to handling sign language lexicon description.

1. Over-Specification

The trouble when filling all parameters with values is that they all inherit the same status. Yet often, some are crucial to the sign in that changing them would destroy the whole sign, whereas others are only given so as to enable, say, a signing avatar to perform the target sign but could well be specified differently. For instance, the palms of both hands in the sign [WHAT]$_{\text{LSF}}$ need be horizontal and facing up, but the fingers may point to anywhere away the signer's body (fig. 1). Actually, the direction they point to may even vary through time, as signers usually prefer to rotate around the wrist or elbow rather than around the shoulder. With a HamNoSys notation, both "fingext" orientations *out* and *out-left* (for a strong hand on the right-hand side) would define the [WHAT]$_{\text{LSF}}$ sign properly, but one has to be chosen.

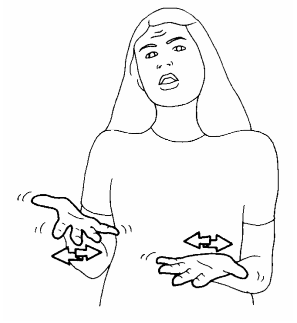

Figure 1: [WHAT]$_{\text{LSF}}$ (Moody, 1986)

The recent addition of the "..." subscript operator in HamNoSys v4 allows to "soften" a value and change it to a somewhat fuzzier specification. That is, used with our example, turn the fingext *out* value into something like "*out* or *out-right* or *out-left*". However, nothing precisely defines this operator, and applying it to the fingext *out* value will also make valid values like *out-up* and *out-down*, which we do not want.

The source of the problem above is that the fingext direction was "hard-wired" to a particular value, and then softened. Instead of over-specifying and merely stating what can be approximated, we suggest that the sign contents should be constrained enough to define the target sign, but that whatever is not necessary be banned from its description. On our example, setting the palm plane normal to an upright vector is enough about the hand's orientation in [WHAT]$_{\text{LSF}}$.

2. Parameter Dependencies

Secondly, parameter models consider the parameters separately and each of them is assigned a distinct value, with no regard for other parameters. In computer science terms, none of these assignments is in the scope of another, so each of them could be carried out in a parallel way, i.e. all at once and independently. Though, this does not account for inter-parameter dependencies, such as that in [DESK]$_{\text{LSF}}$ (fig. 2). The strong hand movement depends on the fingext direction (in HNS terms) of the weak hand, whichever is chosen in the first place.

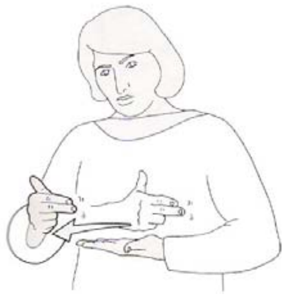

Figure 2: [DESK]$_{\text{LSF}}$ (Moody, 1986)

The issue of parameter dependencies was already addressed and partly resolved with the new HamNoSys "~" subscript operator. It is applicable to palm orientation or fingext direction to make it relative to the path of the corresponding hand. It allows descriptions such as that of $[\text{BALL}]_{\text{DGS}}$, making palm orientation relative to its path on each hand. In $[\text{DESK}]_{\text{LSF}}$ however, the dependency is not one of a hand orientation on its path, but that of one hand's path on the other's orientation.

Moreover, two different parameters could well depend on a common non-parameter object, such as in $[\text{BUILDING}]_{\text{LSF}}$ (fig. 3). The strong hand moves along and close to a line, say *L*. Its palm is constantly facing *L* and the weak hand's location and orientation is defined as being symmetric to those of the strong hand's, with respect to *L*. Both location and palm orientation of both hands depend on the same line *L*. Although *L* is obviously crucial to the sign as a great part of the description depends on it, no parameter (in Stokoe's sense) is ever *equal* to *L*, which is why we call *L* a non-parameter common dependency.

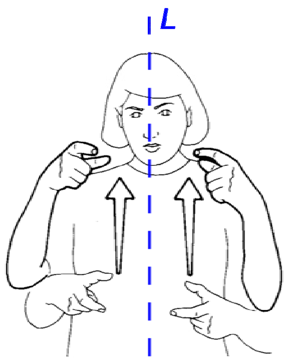

Figure 3: $[\text{BUILDING}]_{\text{LSF}}$ (Moody, 1986)

To account for the two cases stated above, we claim that any part of a sign description should be allowed to make use of other parts, even of the same description. This way, internal dependencies become part of the description.

3. Iconic Structures

Above all, using C. Cuxac's theory (2000) of iconicity as a framework for ours, it has become obvious that the many possible context influences cannot be ignored while modelling lexical description. A great part of sign languages' beauty and power in concision comes from the potential for signs to be altered according to the context in which they are used, thereby switching discourse from a conventional sign flow to *highly iconic structures* (HISs). For instance, the sole sign $[\text{BOX}]_{\text{LSF}}$ can be used to sign the phrase "large box" in LSF, only the distances between the hands will be greater than the ones involved in the plain conventional $[\text{BOX}]_{\text{LSF}}$ sign (plus the signer will probably also puff his cheeks and raise his elbows).

There are many forms of iconicity in SLs: size&shape transfers, personal/situational transfers, use of time lines... Formalising such features for automatic sign generation is not trivial. Some work has been initiated with the ViSiCAST project to include use of proforms and signing space in particular (Hanke *et al*, 2002), but we found nothing close to the richness emphasised in (Cuxac, 2000). An HIS can not only alter the location or the hand shape involved in a sign, but also a path, a direction, eye gaze, etc. Virtually, anything can be acted upon, and these actions being commonplace in SL, we claim a description model should allow signs to behave accordingly. Back to the example above, describing $[\text{BOX}]_{\text{LSF}}$ without making the distance between the hands responsive to the contextual size weakens the sign's re-usability.

4. A Geometrical Approach to Descriptions

We are now ready to outline a proposal for a new sign description model whose aim is to make for the three main problems we see with present parametric models, stated above and summarized below :

- unnecessary parts should not appear in a description;
- the different parts should be able to refer to one another;
- descriptions should be made flexible enough to be responsive to context influences.

Specifying What is Needed and Allowing Internal Dependencies

We handle the first two points using a statement-based language, each of which is either a build statement (B-statement) or a constraint statement (C-statement). B-statements are used to build objects like points, vectors or planes that can be referred to in subsequent statements. C-statements serve the main point: a C-statement either assigns a value to an existing object or adds a constraint to it. Constraints are either applied to an object itself or to one of its "slots" if it has any. A slot is a constituent of an object that accepts geometrical constraints but may remain unmentioned. For example, eyebrows can be set to frown in a sign *S* by slotting a value in the appropriate slot, denoted `S.eyebrows`. Yet in other signs the eyebrows can stay unspecified, and indeed they often do.

The syntax used for C-statements is close to that used in mathematical definitions of geometrical figures (Filhol, 2006). For example, the following C-statement sets a correct orientation (the `ori` slot of the hand) for the strong hand (the `shand` slot of the sign) of a sign `S` by constraining its palm (the `palm` slot of the orientation) to be orthogonal (the "`_|_`" operator) to a direction pointing up (`Up` is a constant) :

```
S.shand.ori.palm _|_ Up
```

The syntax used for B-statements resembles that of variable declarations in most programming languages, i.e. a type keyword and an identifier. Here is a B-statement that creates a plane named `P`:

```
PLANE P
```

Most probably, C-statements enrolling `P` will follow, in order to constrain it and use it afterwards as an internal dependency. For example, to make it horizontal:

```
P _|_ Up
```

Note: Parsing such an input will require some conflict-checking, as two contradictory C-statements applied to an object should be rejected. Though we shall not deal with this issue here.

With this description language, sign descriptions can be specified as much – or indeed as little – as wanted, which tackles the first drawback underlined in part 1. Moreover, each part of a sign description can make use of any other part, provided the latter has been defined beforehand. Thus, contrary to parametric models, value assignments are no more paralleled but made sequential, and values are not only chosen from a fixed set but may depend on intermediate objects (there again provided they were built earlier on) if any are needed. An acyclic dependency graph can then be associated with the description, which represents the description's internal dependencies.

Iconicity in Descriptions

Although no implementation has been done on this issue so far, it has always been regarded as a necessary prospect in the design of our description model. Here is how we will extend the given language to handling iconicity in sign descriptions.

Enabling iconicity in signs can be done by extending the language with a new type of reference. Every time a value or an object is expected in a statement, a call to a context element can be placed instead. For instance, instead of specifying an arbitrary distance between the hands' positions in the description for $[\text{BOX}]_{\text{LSF}}$, we may refer to an external reference called *size*. This way, whenever the description is used to perform the sign in discourse (i.e. in context), it can be tagged with a size attribute, so that the distances are altered accordingly, with no extra rule about how to sign "big box" or "small box".

This brings us to extend the dependency graph to external nodes, in the sense that some of the values within the description will depend on values that are "outside" the lexeme itself. In fact, they are to be found in (or given by) the context/syntactic level.

More generally speaking, this comes down to including semantic information in the lexical units being described. Indeed, it is a reasonable hypothesis that the list of external dependencies relates the cognitive type of the sign's concept. E.g. $[\text{BUILDING}]_{\text{LSF}}$ will at least have the following external dependencies : height and width and situation in signing space. The results we have started to collect from our study of the French conventional lexicon go to show that a lot of signs denoting concrete objects have the same physical dependencies, namely size and location.

5. Full Example

Here is a full example of a description for $[\text{BUILDING}]_{\text{LSF}}$, drawn in fig. 3 further up. Figure 4 illustrates the various objects built within. External dependencies labels are between curly brackets; the outfix `|x|` operation stands for the length of the argument vector `x`; infix `/\` is the vector product operator.

```
1.  SIGN S

2.  LINE L
3.  L // Up
4.  L THRU {Loc}

5.  POINT M
6.  VECTOR V
7.  V = Vect({Loc}, M)
8.  V _|_ L
9.  |V| = {Size}

10. S.shand.config = "BSL C"
11. S.shand.ori.palm = -V
12. S.shand.ori.fingext = Up /\ V
13. S.shand.traj.start = M
14. S.shand.traj.mvt = {Height}*Up

15. S.whand SYM S.shand WRT L

16. REGISTER S "building"
```

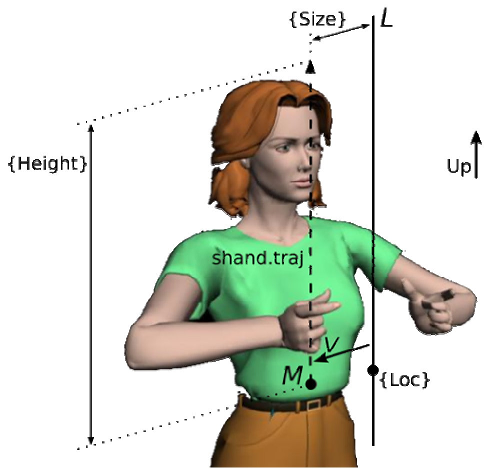

Figure 4: Objects involved in description of $[\text{BUILDING}]_{\text{LSF}}$ below

Line 15 indicates that the weak hand must be symmetric to the strong hand with respect to line *L*. We give it here as an example of the type of C-statement the model might end up with. It actually means that:

- configurations are identical;
- locations verify the given symmetry;
- palm fingext vectors are identical;
- palm normal vectors verify the symmetry.

Hence, line 15 really is a short for:

```
15a. S.whand.config =
          S.shand.config
15b. S.whand.loc SYM
          S.shand.loc WRT L
15c. S.whand.ori.fingext =
          S.shand.ori.fingext
15d. S.whand.ori.palm = V
```

However, we are not yet able to tell whether hand symmetries all behave this way, whatever the sign being described. The only genuine symmetry related in this statement is the one that applies to the hand locations (see line 15b). It may indeed turn out, say, that both hands of a two-hand sign where locations are symmetric along a line have the same normal vector. Please note that the description language is still under development.

6. Conclusion

What we have outlined here is a new way of addressing the description of sign language lexicon units. Instead of merely giving independent values to a given set of parameters, it is based on sequences of constraint statements, which unlike previous models make use of internal dependencies between the elements of the descriptions. Consequently, all the units described do not necessarily mention the same information, but rather each description only states what is needed.

To assess this **geometrical and sequential** approach, we are planning on describing signs on a larger scale. We believe that the flexibility of the suggested language itself will make it easy to cope with many types of constraints, if more are needed. A practical concern in the design of this model is also to limit the number of possible descriptions for a given aspect of a sign, as the fewer there is, the more sign descriptions will look alike, and the more useful the model becomes as to categorize the signs with respect to their descriptions' (or their dependency graphs') layout.

7. References

Braffort, A. (1996). Reconnaissance et compréhension de gestes – Application à la langue des signes. Thèse de doctorat. Université d'Orsay.

Cuxac, C. (2000). La Langue des Signes Française – Les voies de l'iconicité. Faits de Langue n°15-16. Ophrys.

Filhol, M. (2006). Une approche géométrique pour une modélisation des lexiques en langues signées. In proceedings of TALN-Recital 2006, to appear.

Hanke, T. (1989). HamNoSys – An introductory guide. Signum Press, Hamburg.

Hanke T. *et al* (2002). ViSiCAST deliverable D5-1: interface definitions. ViSiCAST project report. `http://www.visicast.co.uk`

Hanke, T. (2004). Towards Sign Language resources – Extending an HPSG lexicon for German Sign Language from empirical data. TISLR 8, Barcelona.

Moody, B. (1986). La Langue des Signes – Dictionnaire bilingue élémentaire. IVT, Paris.