ViSiCAST Deliverable D5-3: Proto-text-to-sign notation

Project Number:	IST-1999-10500
Project Title:	ViSiCAST
Virtual Signing: Capture, Animation, Storage and Transmission	
Deliverable Type:	Int (Internal)

Deliverable Number:	D5-3
Contractual Date of Delivery:	June 2001
Actual Date of Delivery:	August 2001
Title of Deliverable:	Proto-text-to-sign notation
Work-Package contributing to the Deliverable:	
	WP 5 (Language and Notation)
Nature of the Deliverable:	PR (Prototype)
Author(s):	Ralph Elliott, Thomas Hanke, Ian Marshall, Constanze Schmaling, Eva Safar, Margriet Verlinden

Abstract:

This deliverable describes in detail the provisional route from English text to Sign Language oriented notations (HamNoSys and SiGML). In the deliverable, we illustrate the different steps in the procedure: first, the conversion of isolated English sentences to suitable intermediate representation, based on Discourse Representation Structures (DRSs); second, the provisional sign language synthesis for a suitable lexicon, incorporating grammar rules which are representative of the issues that need to be addressed in generating natural sign language structures (rather than sign supported transliterations). The lexicon and the grammar for each of the three target languages (British Sign Language, German Sign Language, Dutch Sign Language) are modelled in a Head-Driven Phrase Structure Grammar (HPSG) fashion.

Table of Contents

1 Introduction

The aim of deliverable D5-3 is to demonstrate the provisional route from English text to sign language notation, specifically demonstrating the feasibility of

- conversion of isolated English sentences to a suitable intermediate representation (based on Discourse Representation Structures DRSs);
- provisional sign language synthesis for a suitable lexicon, incorporating grammar rules which are representative of the issues which need to be addressed in generating natural sign language structures (rather than sign supported transliterations). The lexicon and the grammar for each of the three target languages (BSL: British Sign Language, DGS: German Sign Language, and NGT: Dutch Sign Language) are modelled in a Head-Driven Phrase Structure Grammar (HPSG) fashion.

The prototype described here successfully demonstrates this provisional route from English text to sign language oriented notations (HamNoSys: Hamburg Notation System / SiGML: Sign Gesture Markup Language). At a later point of time, it will be combined with work from Visicast WP4 to give a complete picture of the translation from English into sign language by rendering the output into high-quality sign language animation. At that point of time, the appropriateness of the whole process – and especially the grammatical and lexical correctness of the generation side – will be and can be assessed by native signers without a profound background in sign linguistics. As a consequence, user evaluation of the work described here can start with that combination.

Parts of the work described here will also form the basis for future work in WP2 in the context of WWW pages in sign language. Even a complete route from English to sign language without user interaction would not be an ideal solution for every potential page designer. Instead, native signers might prefer a more or less monolingual approach. In the light of the fact that a written form of sign language does not exist, a gloss-based phrase description, with the HPSG system deriving a correctly detailed sign language sentence seems very promising. (Cf. introduction of deliverable D5-1.)

The least researched area of concern – sign language structure – has sought to address significant characteristics of sign languages and successfully demonstrates synthesis within HPSG of a number of these issues. As work in progress whose baseline starting point is very low, the progress to date as reported here is a significant contribution to sign language research and is indicative of the potential of developments over the forthcoming year and a half.

The original description of the deliverable stated that this would incorporate a lexicon of 500 signs. As the iterative nature of the enterprise became apparent this was scaled down to reduce the volume of modifications within a development lexicon during initial stages of investigation when the lexicon structure is least stable. As a consequence, a rather limited domain was selected for the prototype. On the other side, it seemed appropriate to include coverage of important sign language specific phenomena such as classifier verbs which were planned to be addressed only after this deliverable.

The original description listed a number of English grammatical features which would be converted into the semantic representation – active, passive, declarative, interrogative and imperative sentence types involving adjectival and relative clauses. These sentence types and grammatical structures (plus additional ones) are appropriately converted into the DRS as described below. In particular the example sentences within this deliverable exercise the system's initial abilities to handle plurals and locative phrases which were originally scheduled for investigation within the second 18 months of the project.

The work described here builds upon results from deliverables D5-1 (covering e.g. DRS-based intermediate representation and HamNoSys) and D5-2 (SiGML).

2 Approach

2.1 Prototype Component Architecture

The prototype consists of four modules that form a simple data pipeline. The integration of user-interaction planned for D5-4 will considerably change the data flow between the modules, but the functionality the modules implement will remain the same:

1. English to Semantics Intermediate Representation.
2. Semantics Intermediate Representation to the internal semantics model of the HPSG environment used.
3. HPSG generation taking a semantic form in the internal semantics representation and resulting in a tree structure, the main component of which is a sequence of HamNoSys notations. This component roughly corresponds to the Formulator module in Levelt's (1989) language production model. It is, however, much simpler than that as there are no feedback loops between subcomponents as suggested by Levelt.
4. Conversion of the tree structure into SiGML descriptions.

Due to reasons explained later in the text, two different HPSG systems are in use. As a consequence, there are also two different versions of the second and the fourth module.

2.2 Restrictions

In order to keep a language engineering task manageable, it is common practice to define restrictions on the semantic and lexical domain as well as on which part of the language grammar shall be represented by the grammar fragment to be developed. This is even more necessary when dealing with languages without published lexicons and grammars, with the consequence that a large part of the effort lies in building and then verifying the grammar writer's hypotheses. For this deliverable, we have set up a very limited domain, to be described below. The grammar fragment, however, was chosen to include many sign language specific phenomena. Specific restrictions will be detailed in the text. The most important, however, is that we limit generation to single sentences, resulting in a very limited need for resolution of anaphorical references. Second, the current implementation makes certain assumptions about persons in the discourse that are no longer valid if switching between 'real space' and stage signing space can occur (Liddell 1995). It is planned to install a signing space planner later in the project which will take care of these restrictions.

3 English to Semantics Intermediate Representation

3.1 English to Syntactic Representation

Syntactic analysis is undertaken using the Carnegie Mellon University (CMU) parser (Sleator 1991). The CMU parser is robust and covers a significant proportion of English linguistic phenomena. Part of this grammar is a dictionary that defines the links which a word must make to other words to be incorporated within a sentence. The parser's output is a set of links – a linkage – for a sentence, or more generally, several linkages, corresponding to alternative possible parses. Selection of an appropriate parse is supported by an interface to the possible linkage structures.

3.2 CMU Linkage to DRT[1]

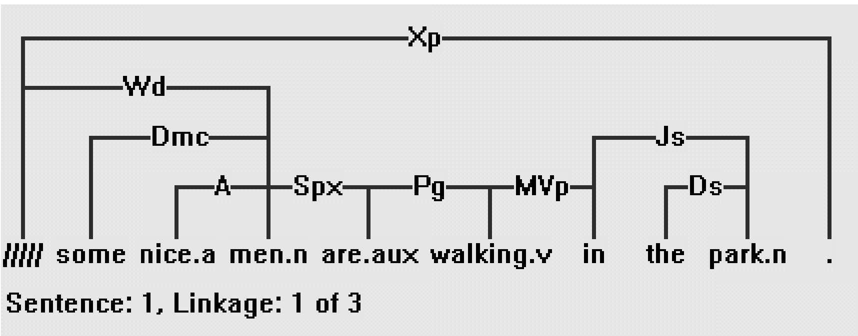
The approach to English to sign language translation is based upon use of Discourse Representation Theory (DRT, Kamp/Reyle 1993) for the intermediate representation of meaning. DRT was chosen as the underlying theory because it decomposes linguistic phenomena into atomic meaning components (propositions with arguments), and hence allows isolation of tense/aspect and anaphoric connections that are realised in different sign language grammatical constructs or modalities. The definition of Discourse Representation Structures (DRSs) as described in Kamp/Reyle (1993) is modified to achieve a more sign language oriented representation that subsequently supports a more direct mapping into a sign language grammar.

A DRS is a two-part-construction involving a list of variables (denoting the nominal discourse referents) and conditions (a collection of propositions which capture the semantics of the discourse). The translation from a CMU linkage to its DRS representation is implemented in Prolog.

In Kamp/Reyle (1993) only event propositions are labelled for use as arguments with temporal predicates. This has been extended by introducing an ontology for all DRS propositions. This allows the possibility of [attr1:big(X), attr2:very(attr1)] to handle adverbial modifiers. In BSL facial expressions convey intensity and hence may be synthesised from such higher order predicates.

In the following example, the process of Discourse Representation Structure (DRS) creation is illustrated. For (1) the CMU parser produces the linkage illustrated below.

```
(1)    Some nice men are walking in the park.
```



The linkage's links are ordered on a fixed preferential basis according to their start and end positions, and irrelevant links are deleted. A link dictionary maps each link type to a lambda-expression DRS definition (lambda-DRS). These are lambda-expressions which may contain embedded DRSs or may take these as arguments. These lambda-DRSs are concatenated to form a sequence of unevaluated lambda-expressions, which is later reduced using functional application (beta-reduction). At the end of the process, when no further beta-conversions can be performed, a merge operation is carried out to build the complex DRS for the sentence as in Blackburn/Bos (1999). The merge operation combines two DRSs by taking the union of the two universes and the conditions (Bos et al. 1994).

---

[1] This section is extracted with some modification from Safar/Marshall (2001).

Example 2 is the lambda expression associated with a 'Pg' link whose right end identifies verbs. *Label* indicates the kind of proposition, i.e. for event propositions *e(Number1)* and for temporal propositions *t(Number2)*, where Number1 and Number2 are unique numbers. *Verb* stands for the verb-stem of the main verb identified by the link.

```
(2)    Pg ---> lambda(X, drs([],[Label1:when(Label2),Tempus,Label2:Verb(X)]))
```

The determination of valid verb stems is achieved using WordNet (Miller et al. 1993) dictionary files. Similarly, morphologically available temporal information is extracted from auxiliaries and verbs (*Tempus*) and is identified by the temporal label (*t*) associated with the proposition as in Example 3:

```
(3)    t(Number)=now, t(Number)=continuous
```

A context free Prolog Definite Clause Grammar (DCG) is used to process the linkage. Example 4 illustrates that the lambda-DRS for *np1* is the concatenation of lambda-DRS for *det* applied to the lambda-DRS for *n2*.

```
(4)    np1(Sent, Det@Noun) ----> det(Sent,Det), n2(Sent,Noun).
```

The link dictionary definitions for determiners and nouns provide lambda-DRSs, which when reduced produce the lambda-DRS in Example 5.

```
(5)    drs([v(1)], [a(1):park(v(1))])
```

The link dictionary entries and the partially evaluated lambda-DRSs can make explicit statements for merging, such as Example 6 for the phrase "are walking in the park".

```
(6)    merge(drs([], [t(0):when(e(0)), t(0)=cont,t(0)=now,e(0):walk(v(0))]),
                 drs([v(1)],[a(1):park(v(1)), l(0):in(e(0), v(1))])))
```

The final DRS is shown in Example 7.

```
(7)    sentence1
       [v(0), v(1)]
       [q(0):exists(v(0))
       attr(0):nice(v(0))
       a(0):men(v(0))
       t(0):when(e(0))
       t(0)=cont
       t(0)=now
       e(0):walk(v(0))
       a(1):park(v(1))
       l(0):in(e(0), v(1))]
```

Though theoretically this is sufficient to implement DRS creation, link dictionary entries can become complicated. Therefore, some DCG grammatical productions contain semantic actions to apply intermediate beta-reduction or merge operations to obtain subexpressions of the required form. This does not violate the compositionality principle and is beneficial for the current application in practical development of link definitions.

```
(8)    modp(Sent,Lab,PP3,Allvalues) --> mod(Sent,Lab,PP,Value),
modp(Sent,Lab,PP2,Value2),
       {mergeDrs(merge(PP,PP2),PP3),append([Value],[Value2],Allvalues)}.
```

Example 8 illustrates the DCG production *modp* which permits a number of verb modifying phrases to be accumulated into a collective partial DRS before later reduction with the verb. The embedded semantic action allows the merge operation to be used to simplify partial DRSs during conversion rather than only at the end.

An alternative rationalisation of the development of an appropriate link dictionary and account of the DRS generation is presented in Marshall/Safar (2001).

Further examples of generated DRS structures are illustrated in the Appendix.

The DRS structure as described however facilitates generation of a flat HPSG semantic structure appropriate for Lingo and through the labelling of propositions for the nested HPSG structure appropriate for ALE (discussed below). Currently the implementation processes approximately the 50% most common CMU link types supporting the constructions planned for this stage of the project.

4 Grammar Formalism

HPSG is the latest framework in the rather short tradition of unification-based grammar theories. With its constrained-based architecture and its strict lexicalism, it differs significantly from Transformational Grammar models although many ideas from Transformational Grammar have been taken over into HPSG.[2]

For most linguists, the choice of a grammar framework they want to work in has paradigmatic status. This most certainly is also the case for us, even if we think that we can give good reasons for our decision in favour of Head-Driven Phrase Structure Grammar:

- HPSG has been used in a number of large projects over the last years, e.g. Verbmobil/Lingo in Germany and the United States (cf. Wahlster (ed.) 2000 and Copestake/Flickinger 2000; see also the survey done by Uszkoreit et al. 1996).
- As a consequence, tool support for HPSG grammar development is available, and the computational model of these tools is well researched. Recent research (cf. e.g. Callmeier 2000) shows that HPSG frameworks can meet efficiency measures as required by interactive systems.
- Project members had some working experience in HPSG modelling.
- HPSG seemed most promising in the discussion how to handle modality-specific problems encountered when working in sign languages.
- Based on the assumption that sign languages are relatively close in their grammars, a lexicalist approach can ease the development for three target languages in parallel: The general grammatical principles are expected to be shared between the three languages, and it is up to the lexicon writer for the individual language to make use of all principles or only some.

With a substantial part of the work described here focussing on language modelling rather than directly on translation issues, we gave preference to HPSG over MT specific formalisms as used in sign language context, e.g. by the ZARDOZ project (cf. Conway/Veale 1994, Veale/Conway/Collins 1998).

In the sign language literature, HPSG has not yet been used very much. In fact, to our knowledge only one group of authors works within the HPSG framework (cf. Cormier/Wechsler/Meier 1999).

Obviously, when aiming at modelling part of the communicative act with a signer, it is an essential question what part of the communicative act is within the language discussed. For spoken language, the border is, even if most difficult in detail, easy to draw in general: Everything not transferred over the acoustic channel definitely is not part of the language, and for most acoustic events it is clear how to cut them into linguistic and extra-linguistic segments.

---

[2] ‘The“ reference book to HPSG is Pollard/Sag (1994). A summary of recent additions can be found in Balari/Dini (eds.) 1998.

Sign languages cotemporally make use of a number of articulators, with the vocal tract being only one of them, but not the dominant one. Most articulators can be used for clearly linguistic as well as for clearly extra-linguistic segments, e.g. manual signs vs. gestures, question facial expression vs. emotional facial expression etc. It is therefore not surprising that there exist a number of proposals severely questioning the standard approach which tries to keep the analogy with spoken languages: Signed utterances consist of a sequence of words, plus some suprasegmental features. (It is on the word level only where we have significant non-sequentiality, e.g. the cotemporality of the articulators, both manual and non-manual.)

Liddell, co-author of one of the most influential papers on sign language phonetics, in his recent works suggests a model in which important classes of signs (agreement verbs) consist of a linguistic, in particular movement shape, and an extra-linguistic component, the indicating gesture directed at a person in 'real space'. The central argument behind this view is the analogue nature of space, allowing an infinite number of directions/targets of signs which therefore cannot constitute phonemes of the language as those need to form an finite set.

From a pragmatic point of view, everything that forms an obligatory and inseparable part of a communicative act conducted in language should be part of the language.

In this context, Liddell's argument means, however, that world knowledge is not only necessary for the correct interpretation of language acts, as is the case with spoken languages, but also for the correct pronunciation of signed language acts.

Ebbinghaus/Heßmann (in prep.) also question the sequence-of-words hypothesis, but from a different perspective: They argue that the mouth component of DGS at least, which undoubtedly stems from spoken language, cannot be treated satisfactorily unless one assumes that two languages are uttered in parallel. It is our assumption, however, that in a lexicalist approach like ours, the phenomena they present in favour of their view can be satisfactorily modelled with the standard assumption that a mouth picture originally derived from spoken language is a non-manual part of a sign language word.

When dealing with languages fundamentally different from English and its close relatives, we cannot expect that the rules and feature sets primarily designed with English in mind would be appropriate. We therefore started with a rather simple HPSG model, namely that introduced by Sag/Wasow (1999), and added features and rules not commonly found as needed. In the following documentation, we try, however, to combine terminology from sign linguistics with terminology from spoken language generation so that it becomes clear which spoken language principles come closest to what we try to model.

Our decision to use traditional part of speech categories for sign language words, even if shared with many sign language researchers, still can be questioned as 'colonialist' and ignorant of the very specific nature of sign languages (cf. e.g. Johnston/Schembri 1999). If we are, however, able to generate signed utterances which are understood by native signers and accepted as well-structured, it would be strong support for this very basic working hypothesis.

5 HPSG Grammar Tools

The grammar and lexical development for the three sign languages have been developed within two HPSG implementations, Lingo for DGS and ALE for BSL and NGT. The choice of framework was mainly determined by installation platforms. Lingo is used on Macintosh computers at IDGS, and ALE is used on Windows PCs at UEA and IvD. The non-availability of each of these on the opposite platform, however, provides an opportunity to investigate the advantages and disadvantages of each in the context of sign language generation. Each implementation however has been constructed around the same generic sign language feature structure.

In addition this has permitted exploration of alternative development practices. Two modes of working were followed:

(a) parse and generate for a sign language (Lingo),

(b) transformation of DRS to the HPSG SEM (ALE).

This has allowed each team to progress independently, though at the current stage of the deliverable only the implementation for the DRS to HPSG SEM structure exists in a mature form for the ALE route (evidenced in the accompanying deliverable demonstration).

5.1 Semantic Models

The ALE (Attribute Logic Engine) written by Carpenter and Penn from version 3.0 onwards provides an HPSG head driven generator (Carpenter/Penn 1999). The BSL/NGT sign language grammars currently use version 3.2 in conjunction with SWI Prolog.

Lingo, written by Copestake, Carroll, Malouf, Oepen and others (cf. Copestake 1999) has been implemented in Lisp, although it does not use the underlying processing system as part of its own processing model as can ALE in using the Prolog resolution machine.

The differences between the semantic models of the systems are mainly due to the different algorithms used. Lingo uses MRS (Minimal Recursive Semantics) with appropriate algorithms (e.g. 'shake and bake', Copestake et al. 1995), while the algorithm of ALE is based on the Semantic Head-Driven (SHD) Generation algorithm of van Noord (1989) and was extended by Shieber et al. (1990).

The following summarises the differences between the two systems.

- The first difference, which is a result of the different generation algorithms in the two systems, is the semantic (SEM) input structure. MRS depends on the use of indices to represent dependencies between the terms in a flat list. ALE, in contrast to this flat list-based representation, needs a nested SEM input structure. Indices, however, are used in the same way as for Lingo to support constructions for agreement and positioning in the grammar.

```
The mug is broken.
sem_x1 := sentence &
SEM [MODE prop,
     INDEX sit-index &  #1,
     RESTR
      <! actthm-predication &
            [RELN break_r,
             ACT #someone,
             THM #1],
         noun-predication &
            [ RELN someone_r,
              INSTANCE #someone],
         noun-predication &
            [RELN mug_r,
             INSTANCE #1]!>
      GLOSS <! "X1" !>]
```

```
The mug is broken
(sent,sem:(index:Ind1,mode:decl,
 restr:[(sit:Ind3,reln:break,act:Ind2,thm:Ind4,
            args:[(index:Ind2,
restr:[(sit:Ind3,reln:someone)]),
             (index:Ind4,count:sg,
             restr:[(sit:Ind3,reln:mug)])])]))
```

- ALE's generation algorithm operates by finding a pivot, which is the lowest node in a derivation tree that has the same semantics as the root. The algorithm is head-driven, which is a natural approach to generation with HPSG. The combination of chain (to generate from the pivot), non-chain rules (for finding the pivot) and recursion result in some differences in grammar rules compared to Lingo. Because of recursion an additional rule which operates as a termination condition (last_complement rule) is needed. However, in ALE only one PRECOMPS and one POSTCOMPS rule is sufficient for any number of complements instead of a collection of rules like unary-PRECOMPS-rule, binary-PRECOMPS-rule, trinary-PRECOMPS-rule, etc. in Lingo).

- ALE's lexical entries function as non-chain rules in the grammar. Therefore, they allow the HamNoSys output to be constructed from the description on the left hand side (LHS) of lexical items, instantiating variables in the LHS via the feature structure by unification. Lingo's framework does not allow the use of the LHS of lexical items to be used as a functional style result, as MRS decouples the lexicon and the semantics. Therefore, the Lingo implementation constructs a hierarchical PHON structure to later construct an appropriate HamNoSys string.

6 Domain Definition

6.1 Choice of Domain

We needed to have a limited domain as our initial domain with a limited set of lexical items which would, however, cover the most important and crucial areas of sign language structure.

We chose a kitchen domain: In the kitchen, we have objects of different sizes which can be handled, manipulated, moved or placed in different ways. We have large (non-moveable) objects which can be located at different locations in space and in/on/under/from/etc. which smaller objects can be placed and moved. This also includes various verbs of movement and location which make use of a large set of classifier handshapes.

We decided not to have food in our kitchen and not to cook as this would have increased the size of our lexicon immensely. It would have also brought up a number of additional grammatical problems. We therefore restricted our corpus to those objects in the kitchen described in detail below.

6.2 Methodology

6.2.1 Data Collection

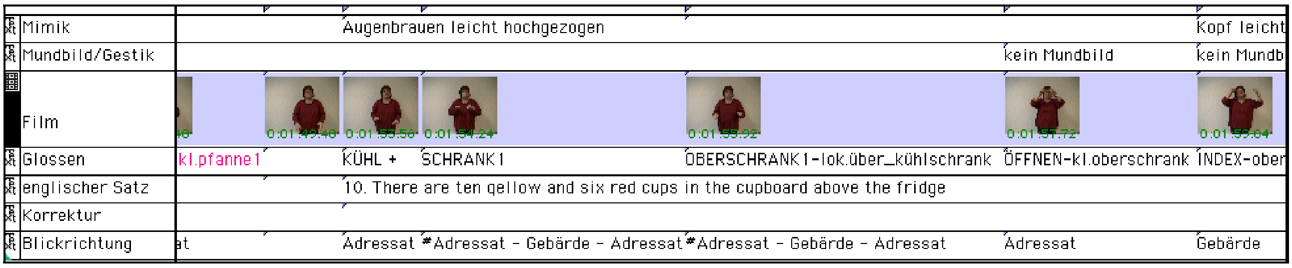
We first set up a list of concepts which were required to cover the most important, critical and characteristic areas of sign language structure (see above).

45 sample sentences were written in English.

For obtaining an equivalent in sign language, native signers were presented with pictorial descriptions of these sentences (e.g. pictures representing the different objects in the kitchen) and were asked to produce sentences in sign language. Their sign language utterances were videotaped and later on glossed, both in English and German.

There are ten yellow and six red cups in the cupboard above the fridge.
Auf dem Regal über dem Kühlschrank sind zehn gelbe und sechs rote Tassen.
KÜHL+SCHRANK1 OBERSCHRANK1-lok.über_kühlschrank ÖFFNEN-kl.oberschrank INDEX-oberschrank FÜR TASSE TELLER1 USW. ÖFFNEN-kl.oberschrank ://SEHEN-oberschrank :// DA-oberschrank ZEHN TASSE GELB SECHS TASSE ROT
KÜHLSCHRANK DARÜBER OBERSCHRANK2 INDEX-im_schrank TASSE ZEHN GELB KLsubst.tasse-lok.im_schrank-pl.versetzt SECHS TASSE ROT KLsubst.tasse-lok.im_schrank-pl.versetzt

In addition, we created a syncWRITER document (cf. Hanke/Prillwitz 1995) which not only included the videos and the gloss transcription but also information on facial expression, eye gaze and mouth patterns.



During the analysis of the signed utterances, we realised that some were too complicated to be dealt with in a first step. Some of the pictorial descriptions were too complex to be retold from memory, and the signers presented too many different versions, and they corrected themselves a number of times.

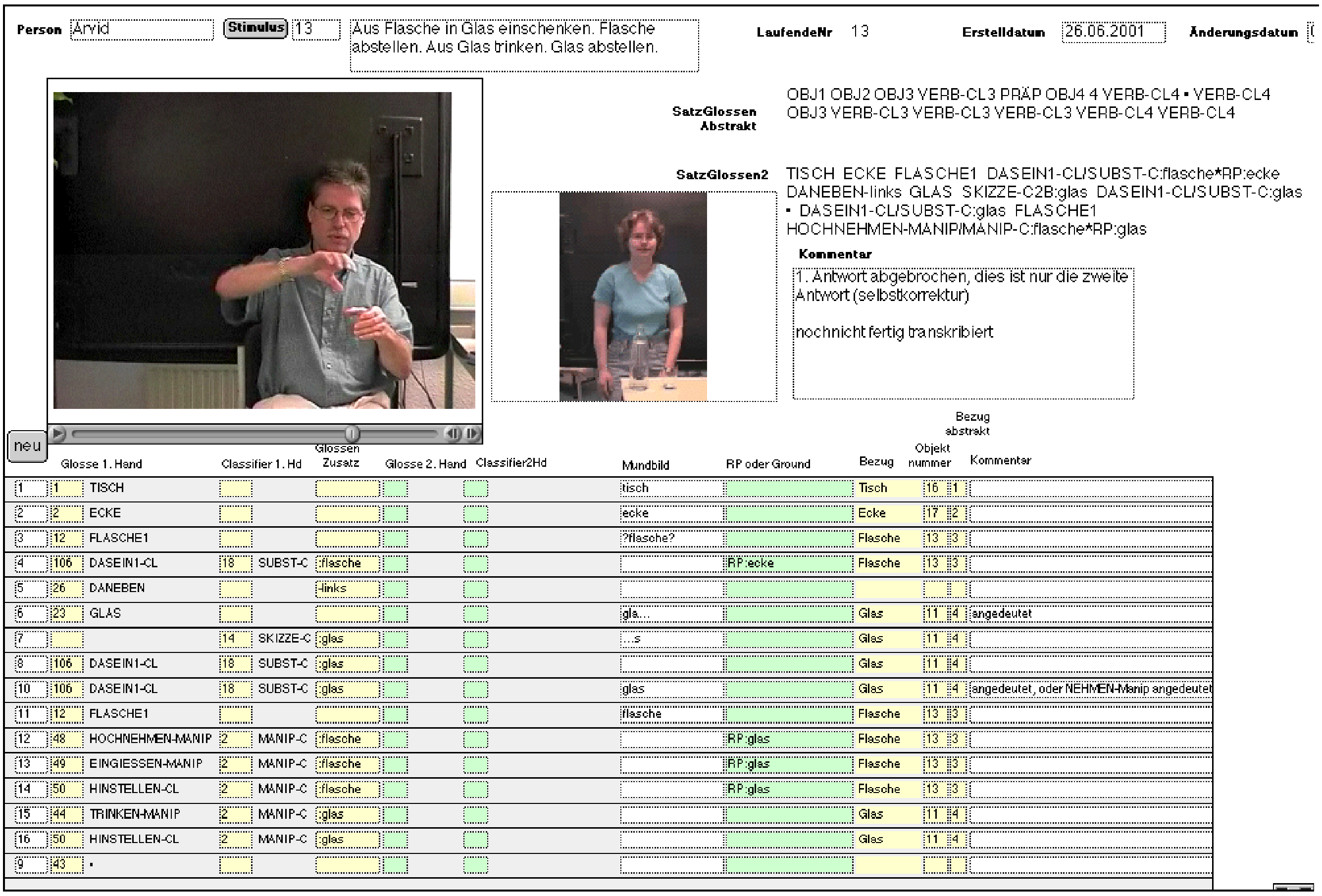
At the same time, we realised that our corpus did not include enough examples for simple phrases as the signers unexpectedly paid too much attention to details. These simple phrases had to be constructed later on and were cross-checked with native signers.

The signers later viewed their videotaped sentences and corrected them or gave additional options for realisation of the same phrase. Problematic or questionable issues were also cross-checked with other native signers.

For further elicitation, we had a second set of approximately 40 sample sentences which were given to five deaf native signers: This time, they were prompted by real-life objects and a person interacting with these objects. This was done in order to

- get more signed examples for the structure of simple sentences; and to
- find out more about the different classifiers for the different objects in the kitchen.

The data was stored in a separate database.



For the BSL data set, two videos of the example sentences were recorded from two different native BSL signers, of different ages and from different regions of England. The first signer was asked to sign the sentences in isolation, but this induced a number of comments regarding the lack of an actual model of a specific kitchen in question creating problems for spatial constructions within the language. The second signer was asked to a sign a story regarding two people in a kitchen during which time the signer developed a model of a kitchen in which the story took place. After this, the signer was then asked to sign the example sentences. Both video tapes have been used as references in the development of the provisional BSL grammar. Although the signers were clearly influenced at times by the structure of the English scripts – and, in the isolated sentences, by the grammatical constraints imposed by use of individual sentences rather than a larger discourse level – their signing did contain many useful examples of BSL grammar.

6.2.2 Data Analysis / Inputting Data Into the HPSG Framework

In a first step, a coding scheme for the lexical items was set up. To verify this scheme, we created a database and filled in a number of lexical items. However, once the feature sets for certain parts of speech turned out to be stable enough, the creation of lexical items was done directly using the HPSG type definition language.

On the phrase structure level, we chose one correct sentence for each example which we wanted to be able to parse and generate, even if several realisations were correct and possible.

6.3 Concepts to be Described

6.3.1 Objects

In our lexicon, we have three groups of objects:

- Small moveable objects which can be placed onto/into other (larger) objects – we have called them 'objects'. In our kitchen, these are: *mug, cup, glass, plate, soupplate, bowl, pan, pot, cover, dishes, coffeemachine; cutlery, spoon, fork, knife, teaspoon.*
- Large objects which cannot be easily handled or moved – we have called them 'location objects': In our domain, this is the kitchen furniture: *table, cupboard, wallcupboard, shelf, stove, oven, fridge, sink, drawer, garbagecan.*
- The personal pronouns: We have only included 1st and 2nd person sg: *I* and *you* (sg.) in our lexicon. We decided not to have 3rd person as this complicates the issue of locatability of persons in space.[3] Currently, only objects can take different locations and can move or be moved between locations and space. 1st and 2nd person, however, have fixed locations – on the body and in space respectively.

6.3.2 Actions

The action concepts were chosen in such a way that they would cover the most crucial and interesting areas in the verb system of sign languages: these include the location of objects in space and of objects in relation to each other; movement of objects in space; handling of different types of objects. All these also involve the incorporation of classifiers (see below).

For the selection of action concepts it was also important to have verbs which take (a) different (number of) arguments and different grammatical features (source, goal, agent, patient, direct object, indirect object).

Our corpus consisted of the following lexical items: *be located, move (sth. from one place to another), take (sth. from somewhere), put (sth. somewhere), give (sth. to sb.), open / close / turn on / turn off / clean (various objects in the kitchen), see, have.*

6.3.3 Properties

These comprise of a set of colours (*red, blue, green, yellow, black, white*), a small set of other adjectives (*new, cold, small, deep*[4]), and the possessive pronouns for 1st and 2nd person singular.

6.3.4 Spatial Relationships

Spatial relationships in our lexicon comprise of: *right, left, above, below, in front of, behind, on, in, into, next to.*

6.3.5 Quantities

Included in our lexicon are the numbers *1–20, some, all.*

---

[3] Several authors have suggested that in sign language one can only differentiate between 1st and non-1st person pronouns – 1st person being performed on or near the body while 2nd person is performed in space (e.g. Engberg-Pedersen 1993, Meier 1990). However, for the generation of signed utterances in our context, locations in space need to be even further differentiated.

[4] Some concepts were only added after we had glossed the videotaped sequences, e.g. MADE-OF-GLASS was not included in our initial list but was added in the adjective section as it was used several times by the signers.

6.3.6 Conjunctions

The object coordination conjunction *and* is included, but no phrase-level conjunction.

6.4 Sample Phrases

The 45 sample sentences were chosen in such a way that they

- included the concepts / lexical items that had been selected (see above);
- represented the different sentence types which we wanted to be able to convert into the semantic representation: declarative, interrogative, and imperative sentence types.

Many sample sentences had a rather complicated structure (e.g. "Please give me the three red pots in the shelf between the oven and the fridge"). On the other hand, we only had few examples of simple declarative sentences such as "I put the knife on the table" or "The glass is on the table". As these 'simple' sentences are already quite complicated due to the spatial configurations involved, we needed to have more data. Therefore, additional sentences of this type were collected in the second set of elicitation data.

6.5 Sign Language Utterances

Even though SOV word order seems to be preferred in sign languages, this is not the 'standard', and many of the signed sample sentences contained other constructions. The native signers accepted different versions for one sentence, and often, individual signers had individual preferences. However, for the parsing and generation of phrases within the HPSG framework, we decided to select the version that would be most appropriate for the formal, information-giving style of a middle-aged signer with a reasonable degree of a 'lay person's' knowledge of her sign language. In this way, some stylistic variations could be reasonably neglected.

7 Grammar Model and Lexicon Model

7.1 Generation and Analysis

Even if only generation is within the main focus of the Visicast project, the Lingo development followed the suggestion by Copestake (1999) and Copestake/Flickinger (2000) as far as possible to write the grammar in a way that it can be used for both analysis and generation. This is not only helpful in testing, debugging, and evaluating the grammar, but it may also contribute to a better user-interface for monolingual description of output in WP2 context as mentioned in the introduction.

In the Lingo implementation, a string of glosses[5] forms the input for analysis and one part of the output for generation. The use of glosses is standard practice in sign language research as it is quite an efficient way to write down sign language. This does not mean, however, that such an approach is undisputed: For example, there are several potential risks connected to using labels from another language. (For an overview of the problems related to working with glosses, cf. Pizzuto/Pietrandrea, in press.)

The ALE implementation employs the pipeline stream described earlier to produce the HPSG SEM structure required to drive HPSG generation.

[5] Glosses – in capitals – are labels using spoken language words, to a large part neglecting the agreement processes which a sign undergoes in order fit into a particular sentence.

7.2 Adaptations of the Standard Model

The standard HPSG rules and feature sets have been designed with mainly English in mind, and many issues were raised when HPSG began to be used for other languages with either relative free word order or, for example, with massive occurrences of pro-drop (cf. Nerbonne/Netter/Pollard (eds.) 1994 on German; and Balari/Dini (eds.) 1998 and Borsley 1987 on Romance). Sign languages are often said to be pro-drop in excess and at the same time have a word order with verb final that is more problematic in HPSG than SVO as found in English, for example.

It is therefore not surprising that many of the rules found in HPSG literature do not apply to sign languages, and need to be extended or replaced. The principles behind these rules, however, remain intact.

7.2.1 Subject and Complements Structure

The standard rules (as found e.g. in Sag/Wasow 1999) for handling complements and subjects are:

- Head-complement rule:

```
phrase[COMPS <>] → H word[COMPS<[1]...[n]>] [1]…[n]
```

- Head-specifier rule:

```
phrase[SPR <>] → [1] H phrase[SPR <[1]>]
```

The rules are clearly based on a relatively strict word order for subject, verb, and objects.

Even if sign languages are often reported to be SOV languages, further research is needed to verify if there is only one underlying word order for each sign language.[6] For the time being, we set up a rule infrastructure allowing a verb to specify any order of its arguments: We postulate PRECOMPS and POSTCOMPS features, but no separate SPR. The head-complement rule then reads as follows:

```
phrase[PRECOMPS <>, POSTCOMPS <>] → [1]…[n] H word[PRECOMPS<[1]...[n]>,
                                     POSTCOMPS <[n+1]…[n+m]>] [n+1]…[n+m]
```

The subject is among the complements and can only be identified by feature-sharing between the COMPS lists and the SEM feature substructure. There has been an extensive discussion whether a separate SUBJ feature is required or not (cf. Borsley 1987). From a practical point of view, we see an advantage in making the subject part of the COMPS lists as the subject is involved in processes such as pro-drop or spatial verb agreement just like the other complements.

7.2.2 Description of the Non-Linear Form

The basic feature structure for the form of a sign is a straightforward HPSG implementation of the structure described in deliverable D5-1. However, that deliverable did not introduce an explicit timing model which was only defined in deliverable D5-2.

As it remains to be seen how satisfactory a model is that treats a signed utterance as a sequence of events with each event specifying the same value for features that hold for the

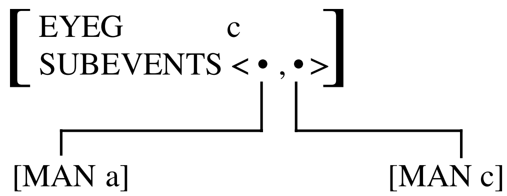
---

[6] One of the examples cited for DGS to show a different word order is a sign often glossed SCH which is usually translated as 'to have/possess'. See, however, Erlenkamp (1999) for an analysis doing away with this problem by treating SCH not as a verb, but as a possessive pronoun. For other sign languages, word order has been reported to depend on the relation of verb and object. E.g. if the verb describes an action that only puts the object into existence, SVO is preferred over SOV.

whole utterance, we introduced a hierarchical model which allowed us more flexibility: The PHON structure has a list of subevents as a feature, allowing the PHON structure of an utterance to be treated as a tree.

The tree structure is reflected in a modified PHON compositionality principle which is respected by the appropriate rules.

$$< \begin{bmatrix} \text{MAN} & \text{a} \\ \text{EYEG} & \text{c} \end{bmatrix} , \begin{bmatrix} \text{MAN} & \text{b} \\ \text{EYEG} & \text{c} \end{bmatrix} >$$



With the implicit assumption that every leaf defaults to empty lists or neutral values and that every other node defaults to the sequence of features specified by the subevents, we have a much more powerful structure than just a list. The future implementation work will show how much this flexibility is explored.

```
phon-struc := feat-struc &
[       MAN    man-struc,
        SHLD   hamnosysstring,
        BODY   body-struc,
        HDMOV  string,
        EYEG   string,
        FACE   face-struc,
        MOUTH  mouth-struc,
        SUBEVENTS *list*].

body-struc := feat-struc &
[       BODY-ROT     string,
        BODY-BEND    string ].

face-struc := feat-struc &
[       BROW   string,
        LID    string,
        NOSE   string ].

mouth-struc := feat-struc &
[       GEST   string,
        PICT   string ].

man-struc := feat-struc &
[       NDH    hamnosysstring,
        HSH    hamnosysstring,
        ORI    ori-struc,
        CONST  hamnosysstring,
        MOV    mov-struc,
        WEAK-HAND weak-hand-involvement,
        REFPT  hamnosysstring].

ori-struc := feat-struc &
[       PLM    hamnosysstring,
        EFD    hamnosysstring ].

mov-struc := feat-struc &
[       REPEAT hamnosysstring ].

path-struc := mov-struc &
[       SRC    path-endpoint,
```

```
       GOL    path-endpoint,
       HSHF   hamnosysstring,
       ORIF   hamnosysstring,
       PTH-MOD hamnosysstring,
       SEC    hamnosysstring,
       FOB    boolean ].

non-path-struc := mov-struc &
[      MLOC   hamnosysstring,
       M-HNS  hamnosysstring,
       ROT    hamnosysstring].
```

The feature WEAK-HAND partially doubles information contained elsewhere in the structure and adds details about weak-hand drop, see below. For the feature REFPT, see the section on reference points below.

7.3 Agreement

Case markers in the traditional sense, as known from many spoken languages, are not found in sign languages. However, there are several mechanisms to show agreement between verbs and the complements:

- So-called agreement verbs unify the spatial loci associated with persons or objects taking a role in an event described by the endpoints of the path movement. A common pattern is a path movement from subject to indirect object, or from subject to direct object if there is no indirect object. This behaviour is modelled by a number of verb lexeme subtypes.
- So-called classifier verbs can pick up certain gestalt features (classifiers) of the direct object or subject and use them as part of their own phonetic realisation.

The notation issues of agreement have been addressed in deliverable D5-1.

Here is an example of the verb GIVE[7], a three-argument verb that falls into both categories and the noun MUG as a potential direct object:

```
give_1 := classifier+actpat-dir-verb-lxm &
[GLOSS <! "GIVE" !>,
 SYN [ PRECOMPS  <!
             phrase & [SYN.HEAD pr-noun,
                       SEM.INDEX #1],
             phrase & [SYN.HEAD pr-noun,
                       SEM.INDEX #2],
             phrase & [SYN.HEAD or-noun &
                           [AGR [ NUM sg,
                                  CL.MANIP <! [CL-NDH #ndh,
                                               CL-HSH #hsh,
                                               CL-ORI #ori,
                                               CL-CONST #const] !> ]],
                       SEM.INDEX #3 ] !>,
       POSTCOMPS <! !> ],
PHON.MAN [ NDH #ndh,
           HSH #hsh,
           ORI #ori,
           CONST #const,
           MOV [ PTH-MOD <!hamarcu,hamsmallmod !>,
                 FOB false ]],
SEM.RESTR <! actpatthm-predication &
             [ACT #1, PAT #2, THM #3, RELN give_r] !> ].

mug_1 := cn-lxm &
[SYN [ HEAD loc-or-noun &
```

[7] Unless indicated otherwise, glosses in this text refer to DGS signs.

```
[AGR.CL [MANIP <! cl-shape-upright-cylinder !>,
         SUBST <! cl-shape-upright-cylinder !> ]]…]].
```

In this example, the first complement is the subject, identified with the source of the movement, the second is the indirect object, identified with the goal of the movement. The direct object brings in the gestalt and also determines the handedness of the sign. There is a second realisation of the sign GIVE which does not incorporate a manipulator classifier (see also below).

7.4 Classifiers

Classifiers comprise of a special set of gestalts (handshape plus orientation plus handedness) which agree with certain properties of their associated objects.

We have grouped the classifiers into five groups[8]:

- Manipulators (manip): The handshape depicts the way a person handles an object; this may include information about size and shape of an object. Manipulators are used e.g. in verbs of repositioning, handling, transfer of ownership.
- Substitutors (subst): The handshape represents an object or part of an object; this may again include information about size and shape of the object. Substitutors are used e.g. in verbs of motion.[9]
- Measurement (meas): For this type of classification, 'unmarked' handshapes are used which give information about the size of an object – but not about its shape.
- Size and shape specifiers (sass): The handshapes give information about the shape of an object, and optionally also about its size. Sasses are used e.g. in object configuration and location descriptions.
- Sketching (sketch): The outline of a 2D-projection of an object is drawn in signing space to indicate its size and shape and possibly location.

Classifiers can be either one-handed or two-handed. In two-handed classifiers, the phonological features are copied onto the nondominant hand; only the kind of symmetry needs to be specified.

The approach we have taken allows for a maximum of one classifier per type for a concept represented by a lexical entry. Some objects, however, can take more than one classifier of a particular type, depending either on the focus of the utterance or on physical details of the object in question, e.g. SPOON (one manipulator classifier when eating with a spoon; different manipulator classifiers when putting a spoon onto a table).

In the kitchen domain, only the small objects can take manipulator and substitutor classifiers. Of course, if one was describing how to furnish a kitchen (from a bird's perspective), both types of classifiers would also be needed for the larger objects.

The classifiers form an appropriate class hierarchy for themselves, i.e. they are not part of speech items.

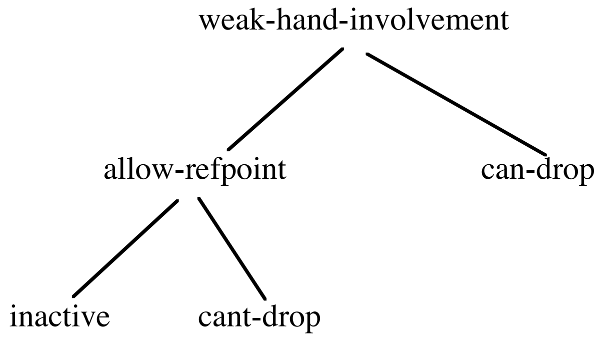
---

[8] In the literature, there have been several attempts to group classifiers (or classifier handshapes) into different categories (see e.g. Schembri 2000).

[9] It is sometimes difficult to decide whether a classifier is used as a manipulator or as a substitutor.

7.5 Weak Drop

It has been reported from a number of sign languages that two-handed signs may be preformed with the dominant hand only if they meet certain criteria, e.g. some symmetry conditions (cf. Battison 1978, among others). However, the rules found in the literature tend to be general or vague, and we therefore decided to include the information on weak-hand drop in the HPSG description of each individual sign. In our grammar, the MAN substructure of PHON contains the feature WEAK-HAND with values from the following hierarchy:



For one-handed signs, this feature is clearly redundant, as the value *inactive* can be assumed.

Two-handed signs can take one of the values *can-drop* or *cant-drop*.

Signs with unspecified handedness, e.g. classifier verbs where handedness is determined by the classifier, can be either *allow-refpoint* or *can-drop*.

The application of this rule is implemented by a pair of HPSG lexical rules (one for compounds, one for non-compound signs).

The rules, however, only determine when weak-drop is permissible for an individual sign. In a generation context, this produces sign sequences with unmotivated changes between performed weak-drops and optional, but unrealised weak drops. This is highly unlikely to occur in real signers' utterances (cf. outlook below). Weak drop is not only phonologically determined but is also determined by the level of discourse. Where figure and ground are both contained in the same sentence (see section below) weak drop may be more likely to occur.

7.6 Reference Points

In signed descriptions of spatial relations, the signer may use a reference point construction: here, the ground is mentioned first. In order to better visualize the spatial relationship, the nondominant hand of the ground sign remains in the location where the movement of the sign ends while the subsequent signs, esp. the figure, are uttered. Obviously, only one hand remains for the following signs. Two-handed signs therefore undergo weak-hand drop where possible. If a two-handed sign cannot weak-drop, the reference point is given up for just that sign, i.e. the nondominant may return to the reference point location after finishing the two-handed sign.

This construction has been modelled by a number of lexical rules that convert simple signs into signs respecting a reference point:

```
simple-sign := lrule-word &
 [ GLOSS #gloss,
   SYN #syn & [ REFPT-IN < > , REFPT-OUT < > ],
```

```
   SEM #sem,
   PHON #phon & [ MAN.REFPT emptyhamnosys ],
   ARGS < lexeme & [ GLOSS #gloss,
                     SYN #syn,
                     SEM #sem,
                     PHON #phon] > ].

one-handed-sign+refpt := lrule-word &
 [ GLOSS #gloss,
   SYN #syn & [ REFPT-IN < #1 > , REFPT-OUT < #1 > ],
   SEM #sem,
   PHON #phon & [ MAN.REFPT #1 ],
   ARGS < lexeme & [ GLOSS #gloss,
                     SYN #syn,
                     SEM #sem,
                     PHON #phon & [MAN one-handed]] > ].

two-handed-sign+refpt := lrule-word &
 [ GLOSS #gloss,
   SYN #syn & [ REFPT-IN < #1 > , REFPT-OUT < #1 > ],
   SEM #sem,
   PHON #phon & [ MAN.REFPT emptyhamnosys ],
   ARGS < lexeme & [ GLOSS #gloss,
                     SYN #syn,
                     SEM #sem,
                     PHON #phon & [MAN.WEAK-HAND cant-drop]] > ].


refpt-establisher := lrule-word &
 [ GLOSS #gloss,
   SYN [HEAD #head,
             REFPT-IN <>,
             REFPT-OUT < #refpt >,
             PRECOMPS #precomps,
             POSTCOMPS #postcomps],
   SEM #sem,
   PHON #phon,
   ARGS < cn-lxm & [ GLOSS #gloss,
                     SYN [HEAD #head & [AGR.REFPT-FORM < #refpt > ],
                          PRECOMPS #precomps,
                          POSTCOMPS #postcomps],
                     SEM #sem,
                     PHON #phon ] > ].
```

A number of features have been built in to support these rules:

- PHON.MAN.REFPT is a HamNoSys string describing the posture of the nondominant hand. If it is non-empty, this is the description of the nondominant hand, i.e. the sign itself needs to be one-handed.[10]
- The features SYN.REFPT-IN and SYN.REFPT-OUT are used to thread the reference point through a chain of signs. (It should be noted that reference points extensions are not in sync with phrase structure boundaries.)
- Reference points currently can only be established by common-noun lexemes. The feature SYN.HEAD.AGR.REFPT-FORM describes the potential reference point a sign can establish.[11]

---

[10] It could be argued that the feature PHON.MAN.NDH could be used to store the posture. However, with distinguishing the two cases, we have the option to implement a different behaviour for the transition between signs.

[11] A sign can define a maximum of one reference point. List notation has been used for technical reasons, namely to be able to test for non-existence.

7.7 Pro-Drop

For many agreement verbs, the agent and patient are not only subject and indirect object, but also source and goal of the path movement. Dropping pronouns in these verbs therefore means no lack of information and not even an increase of ambiguity.

According to our data for DGS, the patient must always be deleted if it is a pronoun. In this case, the agent, if it is also a pronoun, may optionally be deleted. If the patient is a proper noun, however, the pronoun agent must not be deleted. Exceptions to this rule (which are not treated in our grammar) are phrases with emphasis.[12]

We needed different rules for 1st and 2nd person both for PAT- and for ACT-drop.

Illustrated below are the rules for PAT-drop (2nd person), and secondly for ACT-drop (1st person).

```
pat-you-drop := lrule-lexeme &
 [ GLOSS #gloss,
   SYN [ PRECOMPS <! #1, #2 !>,
         POSTCOMPS #post,
         HEAD #head ],
   SEM [ INDEX #index,
         MODE #mode,
         RESTR [ LIST #first-list,
                 LAST #last-list ]],
   PHON #phon,
   ARGS < actpat-dir-verb-lxm &
          [ SYN [ PRECOMPS <! #1,
                        phrase & [ SYN.HEAD.ALLOW-DROP true,
                                   SEM [INDEX #pat,
                                   RESTR <! noun-predication &
                                             [ RELN addressee_r,
                                               INSTANCE #pat ] !>,
                                   RESTR [LIST #middle-list,
                                          LAST #last-list ]]],
                                          #2 !>,
                  POSTCOMPS #post,
                  HEAD #head ],
            SEM [  INDEX #index,
                   MODE #mode,
                   RESTR [ LIST #first-list &
                                [ FIRST patiens-predication & [ PAT #pat ]],
                           LAST #middle-list ]],
            GLOSS #gloss,
            PHON #phon ] > ].

act-I-after-pat-drop := lrule-lexeme &
  [ GLOSS #gloss,
    SYN [ PRECOMPS <! #1 !>,
          POSTCOMPS #post,
          HEAD #head ],
    SEM [ INDEX #index,
          MODE #mode,
          RESTR [ LIST #first-list,
                  LAST #last-list ]],
    PHON #phon,
    ARGS < [ SYN [ PRECOMPS <! phrase & [ SYN.HEAD.ALLOW-DROP true,
                                          SEM [ INDEX #act,
                                          RESTR <! noun-predication &
                                              [ RELN speaker_r,
                                                         INSTANCE #act ] !>,
                                          RESTR [LIST #middle-list,
                                                 LAST #last-list ]]],
                                                 #1 !>,
```

[12] Unlike in NGT where overt expressions of arguments do not appear to have emphatic function.

```
      POSTCOMPS #post,
      HEAD #head & verb ],
 SEM [ INDEX #index,
       MODE #mode,
       RESTR [ LIST #first-list &
                    [ FIRST actpat-predication & [ACT #act]],
                      LAST #middle-list ]],
 GLOSS #gloss,
 PHON #phon ] > ].
```

7.8 Plurality

7.8.1 Nominal Plurality

Not all noun signs can be pluralised; the default value for the feature ALLOW-PL-REPEAT is *false*; i.e. nominal plurality is only possible in noun lexemes whose value for this feature is *true*. There are different mechanisms for pluralisation, e.g. by repeating the noun sign at the same location in space (this may be once or several times); or by repeating the sign at different locations in space (repetition with offset – again, repetition may be once or several times). Presently, we are not specifying the way in which a sign is pluralised; signs are all repeated once.[13]

Lexical rule for pluralisation:

```
cn-plural2 := lrule-lexeme &
 [ GLOSS #gloss,
   SYN #syn & [HEAD.AGR.NUM pl],
   SEM #sem,
   PHON #phon & [MAN.MOV.REPEAT <! hamrepeatfromstart !> ],
   ARGS < cn-lxm & [ SYN #syn & [ALLOW-PL-REPEAT true],
                     SEM #sem,
                     GLOSS #gloss,
                     PHON #phon,
                     ARGS <>] > ].
```

There is a similar rule for the pluralisation of compound lexemes.

In our domain, only object-reference or location-reference nouns can be pluralised; no plural forms for the personal pronouns have been implemented.

'Irregular plurals' can explicitly be entered into the lexicon as plural entities. One such example occurs in the kitchen domain: Instead of using a standard pluralisation of plate, informants consistently used a sign that can best be translated as "pile of plates" unless it was clear from the context that the plates could not be arranged in a pile. As this result might be induced by the elicitation method, further research on this aspect is needed.

7.8.2 Verbal Plurality

Plural forms of verbs are not generally handled at this point of time, they are planned to be included in deliverable D5-4. There are a number of issues that need to be dealt with in the grammar:

- distributive vs. collective plural, each with specific path shapes;
- different types of pluralisation: one or two hands, repetition with/without offset; repetition once/several times.

---

[13] Plurality can also be shown by adding a number sign to the noun sign, or by adding an index sign or a classifier which themselves may be repeated.

Currently, a number of lexical entries differentiate between singular, dual, and plural entities they take as themes. The current implementation of separate entries lends itself to generalisation in a larger lexicon.

7.9 Zero Copula Rules

One of the reasons to question the appropriateness of traditional distinctions between static verbs, nouns and adjectives is the lack of copula and a missing form distinction between attributive and predicative use of adjectives. (E.g. MUG RED is a valid sentence, as is MUG RED $\text{TAKE}_{\text{-I}}$.)

Our data, however, show a difference in distribution for colour adjectives: In attributive use, they can be in pre- or post-noun position whereas RED MUG is not a valid sentence. We therefore assume the existence of zero copula.

As zero copula also occur in a number of spoken languages, there are several suggestions how to handle them in the context of HPSG. Here we follow the suggestion by Sag/Wasow (1999) developed for African American Vernacular English and introduce non-headed rules, e.g.

```
zero-copula-rule-for-adj := binary-nonheaded-sem-comp-rule &
[ SYN [HEAD verb,
       ARG-ST <! !> ],
  SEM [MODE prop,
       INDEX #2],
  H #1 & phrase &
         [SYN.HEAD noun],
  NH1 phrase &
         [SYN [ HEAD adj & [MOD <! #1 !> ]],
          SEM.INDEX #2 ]].
```

These rules avoid the introduction of elements with empty surface form with the payoff of introducing rules with higher language dependency risk than others.

7.10 Coordination Rules

At the moment, we can generate sentences in which two NPs, each consisting of an object-reference noun, are coordinated. The coordination of complex NPs still causes some problems. Coordination of whole phrases each with a different verb as HEAD is not possible yet.

```
or-coordination-rule :=  binary-nonheaded-rule &
[ SYN [ HEAD or-noun &
                    [AGR [NUM pl,
                     COORD true]],
        PRECOMPS  #prec,
       POSTCOMPS #postc ],
  SEM [ MODE #mode,
       INDEX #s,
       RESTR [ LIST [FIRST ref-and-predication  &
                                 [ WHOLE #s,
                                   PARTS <! #t, #u !> ],
                     REST #first],
                     LAST #third ]],
  H phrase & [ SYN [ HEAD or-noun & [AGR.COORD false],
                    PRECOMPS  #prec,
                    POSTCOMPS #postc ],
              SEM [ MODE #mode,
                    INDEX #t,
                    RESTR [LIST #first, LAST #second]]],
  NH1 phrase & [ SYN [ HEAD or-noun & [AGR.COORD false],
                      PRECOMPS  #prec,
                      POSTCOMPS #postc ],
                SEM [ MODE #mode,
                      INDEX #u,
                      RESTR [LIST #second, LAST #third]]]].
```

7.11 Differences between Target Language Models

The feature structures in both systems are very similar, with mostly minor differences. The only difference is the ARGS feature in the SEM, which ALE uses to find the pivot required by the generation algorithm.

Many of the differences between the three target languages are determined by the lexical entries for signs. While BSL adjectives do have a tendency to be predicative, prenominal attributive adjectives seem to be more common than they are in DGS or NGT. This is especially true for colour adjectives which are characteristically prenominally attributive. Hence variation between languages and indeed within a language, i.e. whether a verb has pre-complement or post-complement (or a mixture of the two), and whether an adjective is pre or post modifier of a nominal, are handled by different lexical definitions across the three languages.

In addition, many sign languages exploit notions of classifiers which we believe are catered for adequately across the three languages by appropriate classification of verbs and nouns and association with specific classifiers. Hence, again the lexicalist approach addresses these issues.

As a further example, BSL facial expression and mouth shape have a less direct relationship to the local spoken language than in DGS and (to a lesser extent) NGT. Again, this variation is accommodated to the extent in which features associated with mouthing and facial expressions are utilised within the lexicon.

As these advantages follow from the lexicalist position of a framework such as HPSG, we merely note that the type structure and classification described in the following section has been productive across the three languages.

More fundamental, however, are the differences between the sign languages regarding grammatical features which reflect how a signer is allowed to modify signs and exploit methods of co-reference which are a side-effect of the physical activity of presenting signs in three dimensional space. For example, weak-drop and associated reference points are particularly evidenced in DGS but only to a lesser degree in BSL. In BSL, the use of pointing for co-referencing seems to be preferred, or at least acceptable. E.g.:

```
FRIDGEa loca locb CUPBOARDb OPEN-L-H-DOORb PLEASE PUT-Inb SMALL-BOWLb locb
CUPBOARDc bPUTc

(Where suffixes a,b,c denote specific unique locations in signing space
loca,locb,locc respectively.)
```

However, weak drop in BSL was in evidence, especially when the sign was retained as a topic marker allowing further signs to be directed to the non dominant hand's (ND) handshape.

```
SINKc …  UNCLEAR-SURFACE-OF-WATER [ND:-----; D: LOC down to c] KNIFE

CUPBOARDc, above head [ND: -----; D: LOCc, above head]

COFFEE-MACHINE  [ND: -----; D: SPOON LOCa HAVE]

(Where [ND: hns; D: movement] denotes the holding of the handshape on the
nondominant hand from the preceding sign while the temporally simultaneous
action on the dominant hand occurs.)
```

BSL also does not make use of preposition signs to the extent of other sign languages, relying instead on relative locations of referents.

DGS and NGT appear to have rather stricter rules for the ordering of figure/ground or small/large referents, always preferring larger items to appear first in the sentence. While BSL does follow this pattern, there are many instances where the smaller item may appear before the larger one. This aspect requires further attention to determine if this can be treated lexically or by grammatical rules in the current framework.

These kinds of variations can be catered for by

a. incorporation of specific features within the feature structure which are aimed more specifically at one/two target languages;
b. variations in the grammar rules and principles which determine/constrain how these features are used.

The stylistic preference to use pointing rather than using weak drop in BSL has resulted in under-use of these features in the feature structure in the BSL grammar rules in comparison with the development of the DGS grammar.

In general, however, these language differences and cases of idiosyncracies in the feature structure are relatively few in comparison with the levels of commonality which we feel justifies the original decision of following the HPSG lexicalist approach.

8 Lexicon: Types and Type Hierarchy

8.1 Nouns

Nouns fall into one of three groups:

- object-reference nouns (or-noun);
- location-reference nouns (lr-noun);
- person-reference nouns (pr-nouns): these can either be specific persons (no example in our corpus), or pronouns.

All three can be either

- invariant nouns: they have a fixed phonological form; this includes the pronouns; or
- locatable nouns: their location in space can change according to the context – i.e. it is not specified. The feature locatability is very important in sign languages.

An important feature of sign languages is that some concepts my be represented in a variety of ways, for example: a cupboard may be a large cupboard or a small cupboard, it may be opened with both hands or with one hand only; it may have a handle which one holds horizontally or vertically, etc.

8.2 Compound Nouns

Compound signs can consist of two or optionally more components (the latter do not occur in our corpus). The components can be any type of sign/lexeme, but we currently only have compound nouns.[14] In our corpus, compound signs have two ARGS, whose PHONs are added in the PHON.SUBEVENTS feature.[15]

```
compound-lxm := proto-cn-lxm &
 [  PHON.SUBEVENTS < [MAN #1], [MAN #2] >,
    ARGS < [PHON.MAN #1], [PHON.MAN #2] > ].
```

In the HPSG description, we have two different types of compounds:

[14] It remains to be seen if other real compounds exist which are not nouns.

[15] This has not yet been implemented for BSL as the research on compounding is still on-going.

- Non-headed compounds (no head): They take classifiers independent of the classifiers of their components these have to be specified. In the compound sign COFFEEMACHINE (COFFEE + DRIZZLE), – both components do not have classifiers but the compound has both a manip and subst CL.

```
coffeemachine_1 := compound-lxm &
 [SYN [ HEAD loc-or-noun &
          [AGR.CL [MANIP <! cl-subst-large-round !>,
                          SUBST <! cl-shape-round-twoh !>]]],
  ARGS < cn-lxm & [GLOSS <! "COFFEE" !>],
  cn-lxm & [GLOSS <! "DRIZZLE" !>] >,
  GLOSS <! "COFFEEMACHINE" !>,
  PHON [ MOUTH.PICT "kaffeemaschine",
         MAN.WEAK-HAND cant-drop ],
  SEM.RESTR <! [RELN coffeemachine_r] !> ].
```

- Headed compounds take the classifiers of one of the compound's components. There are two subtypes: head-first, e.g. SOUPPLATE (PLATE + DEEP), and head-final, e.g. FRIDGE (COLD +CUPBOARD).

```
fridge_1 := hdfinal-compound-lxm &
 [     SYN [HEAD loc-or-noun,
            ALLOW-PL-REPEAT true],
       ARGS < adj-lxm & [GLOSS <! "COLD" !>],
        cn-lxm & [GLOSS <! "CUPBOARD" !>] >,
       GLOSS <! "FRIDGE" !>,
       PHON [MOUTH.PICT "kühlschrank",
             MAN.WEAK-HAND cant-drop],
       SEM.RESTR <! [RELN fridge_r] !> ].
```

This approach has the advantage of avoiding redundancy in the lexicon. PHON-features are inherited from the components, whereas other features are defined for the whole compound.[16]

Another reason for using this approach is the partial locatability of some compounds, i.e. in some compounds, only one component can be relocated in space whereas the other cannot.

8.3 Verbs

We differentiate between two types of verb lexemes, those which have a fixed phonological form and those in which single or all features within the PHON structure can change:

- invariant verbs: they have a phonological form which is fully specified and does not change independent of actor, patient, source and goal (e.g. HAVE).
- directional verbs: there are various subtypes:[17]
    - fully directional, i.e. beginning and end of movement are dependent on location of referents in space (e.g. MOVE);
    - partially-directional: these may be
        - end-directional: they have a fixed starting point, but the end of the movement depends on the location of the referent in space (e.g. SEE)
        - beginning-directional (the other way around; no example in our data).

The directional verbs can further be differentiated as to whether or not they can take (incorporate) classifiers:

---

[16] The Lingo version references compounds by means of glosses for technical reasons only; lexeme identifiers would be more appropriate as glosses are not unique.

[17] In NGT, the 'auxiliary' verb OP also behaves like the directional verbs. It has not been implemented in our grammar yet as it can only agree with animate arguments and is therefore optional in the context of our domain with a maximum of two animate discourse referents.

- Some cannot, e.g. SEE.
- Others can have two different forms, one is the ‘neutral’ form with a fixed phonological form; the other can incorporate a classifier and takes as PHON-features those it gets from the respective classifier (e.g. TAKE – see also illustration of GIVE above).

For implications of directionality for PHON see deliverable D5-1, Chapter 5.

8.4 Adjectives

We differentiate between two different types of adjectives with respect to their position of the associated noun:

- Some adjectives may either appear in PRECOMPS or in POSTCOMPS position of the specified noun. In our corpus, this is true for the colour adjectives.
- Others have a fixed position, either always before or after the corresponding noun. In our corpus, we only have adj-post-lxm: always in POSTCOMPS position, e.g. NEW, SMALL.

If there is more than one adjective, they appear in the order in which they are specified in SEM.

8.5 Possessives

Presently, our lexicon only contains the possessive pronouns for 1st and 2nd person sg.: MY and YOUR_{sg} (see above).

8.6 Prepositions

Due to the capability of sign languages to directly show spatial relationships, prepositions as they are found in spoken languages are not always expressed overtly in the same way in sign languages.

We therefore differentiate between two types of prepositional phrases, one with a spatial preposition that is overtly expressed, the other without such a preposition. In the latter case, the semantics is contained in the arrangement and the relationship of objects in space.

8.7 Determiners: Quantifiers

We have two types of quantifiers, those that have singular referents (quant-lxm), in our corpus number 1, and those that have plural referents (quant-non1-lxm), in our corpus number 2–20 and SOME. Note that we have not yet included ALL as this requires a lot of ‘world knowledge’ about the respective objects.

9 Implementation

9.1 Practical Issues in Using the HPSG framework

The CMU to DRS system has been developed as an SWI Prolog implementation with system support via XPCE (now incorporated in SWI Prolog) for developement purposes. XPCE may be used for prototyping some simple GUIs during implementation in the future.

As explained in earlier sections in this report, the implementation is performed for two different configurations. The DGS grammar is implemented on Macintosh computers with

Lingo, the BSL and NGT grammars are implemented on PCs with ALE. The parse/generation software for Lingo and ALE have been written in LISP and Prolog respectively.

As can be concluded from examples in this report, the data structures in the grammar are large and complex (the examples illustrated in the text are only small ones). This holds for HPSG grammars in general, as a consequence of its lexicalist nature. For this reason, a developer's environment is desirable in which the grammar writer can view, test and edit data easily, preferably by direct graphical manipulation instead of typing in commands.

For the PC/ALE/Prolog environment, two very useful environments have been written (Pleuk, cf. Calder 1993, and Hdrug, cf. van Noord 1989). Unfortunately, neither of them can be used currently in the context of this project: Pleuk has not been updated for version 3 of ALE, and currently any system including Hdrug is tied to compiled Sicstus Prolog which then cannot cope with the ALE Visicast grammar.[18]

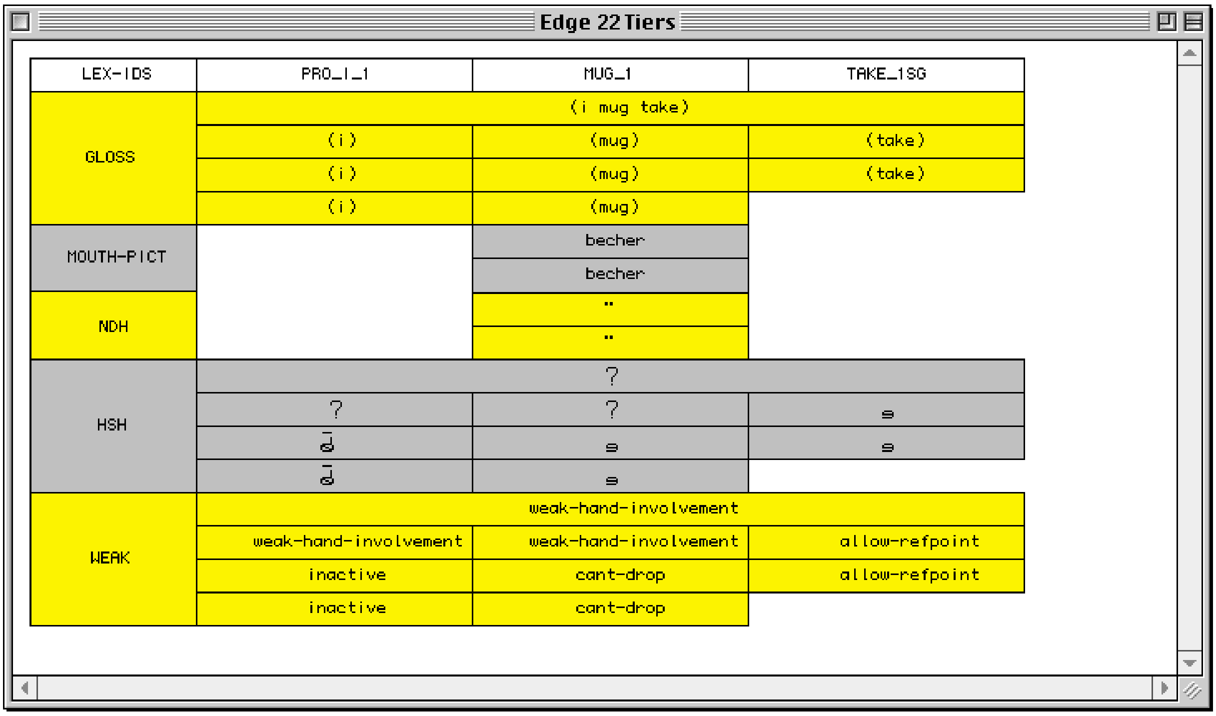
This has been compensated for by integrating automatic testing procedures with development and by modifying some ALE internal predicates to return results rather than print output so that prettyprinting functions tailored to our needs could be implemented.

Use of the SWI ALE version has permitted other minor modifications inside the ALE source code to help development and support the ALE LHS of lexical items exploitation discussed above.

In Lingo, a number of output formats are already available to the user. However, we added a viewer for generation output for the convenience of the sign linguists involved in the project. It displays feature hierarchies in a multi-tier representation researcher are used to from other contexts. Even more important is the display of HamNoSys values with the well-known glyphs instead of textual representations.

As Lingo is provided as LISP source code, the complete integration of the viewer into the Lingo system was possible with reasonable effort.

[18] The ALE-based HPSG system implementation has been developed using SWI Prolog, Sicstus Prolog, XPCE and Hdrug. Currently this implementation is maturing with use of SWI Prolog. Attempts to run ALE with Sicstus Prolog resulted in compiled grammar rules with more than 256 variables due to the size of the Sign language feature structure. These violated a Sicstus system restriction for compiled Prolog. Indications are that this restriction may be removed in further release of Sicstus Prolog, and should this be the case then a faster implementation may result.



9.2 DRT to Lingo/ALE semantics

Sign language generation is achieved by a HPSG generation system, which generates a sign language syntactic form from an underlying HPSG semantic description. Hence the intervening stage converts the DRS representation to the HPSG SEM structure from which synthesis can commence.

This transformation stage consists of two major stages:

(a) re-structuring of the DRS to a more sign language oriented form;

(b) conversion of the subsequent DRS to the HPSG SEM form.

Currently, the former stage performs only minor re-structuring as the initial choice of DRSs as an appropriate representation and the design of the HPSG sem structure to largely mirror the DRS structure results in a conversion which is relatively direct.

The situation where this is less the case is in the use of BSL EXIST_HAVE, where a sentence such as ‘There are knives in the sink’ can be appropriately and preferably signed as SINK HAVE KNIVES, in contrast to the SINK KNIVES THERE/IN. In the former case the arity of the relations in the sign lexicon do not match those in the DRS and this stage reconciles the difference.

The second stage is dependent upon the requirements of the HPSG generation algorithm. The ALE implementation of the generation algorithm is suited to a nested semantic structure which supports synthesis as discussed below/previously. The Lingo implementation of the ‘shake and bake’ algorithm requires a flat structure.

Currently, the more difficult task of conversion to the ALE nested structure has been implemented to support BSL/NGT sign synthesis. This conversion stage is decomposed into three sub-stages:

(1) determination of the sentence type and so the sentence level SEM features can be determined appropriately;

(2) determination of the dominant predicate(s) of the DRS so that this can be treated as the outermost SEM relation;

(3) recursively determine the subordinate-DRS predicates associated with the arguments of this(/these) dominant predicate(s) and generate the appropriate nested SEM structure. During processing of nominals of the DRS form a(n) : pred(X), information from modifier predicates (attr(n):p(X)) and (c(n) = count(X)) is assimilated. (It is envisaged that verb modifying constructions will be determined similarly.)

These stages are marginally complicated by predicates within the DRS (such as the representation of plurality and numeric determiners) which need to be combined with the nested SEM structure of the appropriate nouns as a ‘count’ feature in the SEM.

The design of this conversion stage has to tolerate the obvious revisions which occur in the design of grammars for languages for which there is little published detail. Hence the construction of the HPSG SEM is partly driven by the DRS structure as determined by the sentence level predicates and dominant predicate, but is mainly determined by the structure of the available sign language lexicon. Each lexical item specifies the SEM structure in which it can engage. This includes a relation name (identical to the predicate name of a corresponding predicate in the DRS) and the arguments it requires. Revisions in the definitions of the SEM features of a sign are thus accommodated by dynamic construction of a Sign-SEM dictionary from the available sign lexicon.

The conversion process uses the predicate name within the DRS and its arity to select an appropriate Sign Lexicon SEM feature structure which is then incorporated within the SEM structure under construction. In this way, failure to synthesis a sign language string due to limitations within the lexicon are detectable as failures to construct an appropriate SEM structure from which to begin generation.

The examples in the Appendix illustrate the DRS input and the HPSG SEM output for the ALE implementation. The original intent was to support in this deliverable conversion of DRSs to the HPSG SEM feature for both Lingo and ALE. Initial prototypes were developed for both forms, however the iterative nature of the design of an appropriate feature structure to support sign languages required revisions to both prototypes. For this reason, the ALE implementation was redesigned to be driven mainly by lexical entries’ SEM feature structures, and the Lingo implementation has been delayed to build on the revised ALE implementation.

9.3 Extraction of Features and Conversion into SiGML

The output from the ALE-based translation is a text stream defining a sequence of HamNoSys signs, each of which is represented by a bracketed sequence of symbol names prefixed with a gloss name. The final stage in the translation pipeline is the conversion of this HamNoSys text stream to SiGML, an application of XML (http://www.w3.org/XML) for signing gestures (described in the deliverable D5-2), which form is used as input to the animation tools developed in ViSiCAST WP4. This translation is achieved by a separate software module called HNS2SiGML, which we describe here.

HNS2SiGML operates as a text stream filter, taking as input a textual representation of a sequence of HamNoSys signs, which it converts to the corresponding SiGML document. At present the input is restricted to the manual signing and top-level components of core SiGML, as described in sections 3 and 5 of the deliverable D5-2. At the time of delivery of that report there was a further restriction to HamNoSys 3 input. Now, the majority of the extensions defining (manual) HamNoSys 4, as presented in section 3 of the deliverable D5-1, are also acceptable as input to HNS2SiGML. HNS2SiGML is implemented in Java (http://java.sun.com), using the Antlr parser generation tool (http://www.antlr.org) and the JAXP software for XML processing (http://java.sun.com/xml), which in turn depends on the

Crimson parser and Xalan transformation tool developed under the auspices of the Apache foundation (http://www.apache.org).

The conversion performed by HNS2SiGML is itself implemented as a sequence of transformations applied to each sign in the input. The major stages in this sequence are:

- conversion of input text to a pair of strings defining the gloss name and the HamNoSys text for a given sign;
- parsing of the HamNoSys string to form the corresponding abstract syntax tree (AST);
- traversal of the AST to generate a HamNoSysML (HML) representation of the sign; (HML is another XML application designed to allow a signing gesture to be represented by an XML element structure which corresponds closely, although not isomorphically, to that of the parse tree for the corresponding HamNoSys sign) the resulting structure is represented internally as a DOM tree (http://www.w3.org/DOM);
- transformation of the HML element to the corresponding SiGML element;
- validation check against the SiGML document type definition (DTD);
- output of the SiGML.

The first of these stages accepts two forms of input: one in which each sign is represented as a sequence of symbolic identifiers for HamNoSys symbols (the form which is of interest in the present context), and the other in which each sign is represented directly as (a Unicode encoding of) a string of HamNoSys symbols. The second and third stages (parsing the HamNoSys, and generation of HML from the resulting AST) are implemented using the Antlr parser generation tool for context-free grammars. As the HamNoSys syntax exhibits a number of ambiguities, the syntactic lookahead feature of Antlr is used quite extensively to resolve these. The third stage also tags each sign element with its gloss name. The fourth stage is implemented as an XML transformation defined using the XSLT notation (http://www.w3.org/Style/XSL) and implemented using the Xalan processor. Thus, this conversion stage and the following validation stage, rather than operating at the granularity of the individual sign, are usually applied to an entire HML document representing a sequence of signs.

Examples of the conversion performed by HNS2SiGML, as it applies to individual signed sentences, may be found in the Appendix

9.4 Implementation Differences Induced by the Different Frameworks Used

- ALE provides the same framework for lexical rules as Lingo. In the BSL grammar LHS of lexical entries (terminals) are lists, but lexical rules in ALE assume strings as the words. A remedy to this problem is to implement the equivalent of lexical rules as lexical principles (e.g. for plural) instead. From this it follows that lexical entries for plural are generated dynamically during the process.
- ALE’s macros allow the user to define a description once and then use a shorthand for it in other descriptions. Lingo does not have this facility but makes an extensive use of subtypes in the same fashion.

10 Outlook

10.1 Increasing the Coverage of the Fragment

It is quite natural that we plan to extend the HPSG models for the three target languages by extending both the lexicon and the grammar.

For the current domain, there are a number of lexical items which were included in our initial list of concepts which, however, did not occur in the signed data or which we did not yet incorporate in our HPSG grammar. They will be added in the coming months.

The next step will then be to extend the domain.

A number of grammatical features not or not fully supported have already been indicated in the text. In addition, the following linguistic phenomena need to be covered:

10.1.1 Negation

Negation can be expressed by manual and non-manual signs that can properly be handled in our framework. The combination of negation with quantifiers, however, needs more research before it can be properly addressed in a formal grammar.

10.1.2 Temporal Information

As there is no tense system in the sign languages considered, temporal information is primarily expressed by adverbials. These can be manual signs or non-manuals.

10.1.3 Verb Aspect, Manner and Mood

Many verbs in the three target languages can inflect for aspect (such as durative, habitual, etc.) as well as manner and mood. The ontology of verbs needs to be enlarged to cope with different permissible inflections and their phonetic realisation.

10.1.4 Serial Verbs and Verb Sandwiches

Serial verbs and verb sandwiches are both used in NGT as well as in BSL. They split arguments over two signs, and one of the main questions to be addressed is how to group arguments.

10.1.5 Copy Pronouns

We have not implemented the occurrence of copy pronouns in our grammar yet as they were not used in our sample sentences. They certainly will need to be addressed within a wider domain.

10.2 Changes to the Current Model

10.2.1 Reference Objects

Persons or objects introduced into a sign language discourse are often attributed more properties than can be inferred from the signs they are introduced with, e.g. size of a person or object, location, certain aspects of shape. These attributes are assigned by the signer on the basis of background or default knowledge. It is often crucial, however, that these attributes are kept consistent over the referent's lifetime in discourse. One possibility to model this extra knowledge is the introduction of reference objects (cf. Habel 1986) which have been used in an HPSG framework in the SYNPHONICS project (cf. Abb et al. 1995).

10.2.2 Signing Space Planner

When new referents are introduced into a sign language discourse, they are assigned a location in signing space. Depending on the type of referent and on how it is introduced, the assignment may be to an arbitrary position not conflicting with other referents or in relation to locations for other referents.

It will be the task of a new component in the generation system to model appropriate signing space planning strategies. It will also be the task of this component to monitor the visual prominence of the discourse referents and to block anaphoric reference when switching from ‘real space’ to stage space and vice versa.

It is obvious, however, that an incremental planner cannot achieve optimal results in sign language story telling and comparable discourses where spatial planning might be part of the signer's dramaturgy. Such a model unfortunately lies beyond the scope of this project.

10.3 Ranking of Sentences Generated

The generation process generally produces a number of solutions. If only one result is to be output by the system, a decision has to be taken which one is the best. To a certain degree, this can be achieved by appropriately ordering the rules as well as the entries in the lexicon, an approach currently taken.

However, as there is no possibility to order lexical rules in comparison to grammar rules, and preference of rules might be context-dependent, this approach needs to be replaced by a more flexible solution.

In the context of a semi-automatic translation environment envisioned for the final project result, this is not necessarily an fixed precedence, but could be used on user preferences or corpus data.

In general, with the ALE implementation the current view is to allow the ALE/Prolog top-to-bottom, left-to-right ordering of exploration of the search space to determine the preferred sign language construction in cases where multiple possibilities can be generated. In the case of stylistic variation, this requires identifying one possibility as stylistically preferable and arranging the grammar accordingly. Though there may be occasions where this may be inappropriate, it seems that there are sufficient cases where this seems desirable to explore this approach.

For example, the BSL grammar contains two lexical entries for ‘give’, one which is parameterised by the handshape of the manipulator classifier of the theme and one with a neutral handshape. In cases where the theme has a manipulator classifier this prioritorisation favours use of the former definition.

As a further example, over a longer text, a signer could chose not to locate discourse referents in signing space and always refer to them explicitly. Conversely, having taken the trouble to place discourse referents in signing space, it seems appropriate to allow the grammar to prioritise grammar rules which exploit this. Of course, this assumes an intelligent choice of which referents to place in signing space and which to omit, and this may require user intervention. However, once these decisions have been made, the deterministic ordering of ALE/Prolog should allow appropriate use of prioritised rules.

11 References

Abb, B., C. Günther, M. Herweg, K. Lebeth, C. Maienborn and A. Schopp. 1995. Incremental grammatical encoding - an outline of the Synphonics formulator. In: Günther, C., A. Schopp and S. Ziesche (eds.), *Hamburg working papers on language production IX*. Universität Hamburg, 5-30.

Balari, S. and L. Dini (eds.). 1998. *Romance in HPSG*. Stanford CA: CSLI

Battison, R. 1978. *Lexical borrowing in American Sign Language*. Silver Spring MD: Linstok Pr.

Blackburn, P. and J. Bos. 1999. *Representation and inference for natural language: a first course in computational semantics. Volume II*. http://www.coli.uni-sb.de/~bos/comsem/book1.html

Borsley, R.D. 1987. *Subjects and complements in HPSG. Report CSLI-87-107*. Stanford CA: CSLI.

Bos, J., E. Mastenbroek, S. McGlashan, S. Millies and M. Pinkal. 1994. *A compositional DRS-based formalism for NLP applications. Report 59*. Universität des Saarlandes Saarbrücken.

Calder, J. 1993. Graphical Interaction with Constraint-Based Grammars. In: *Proceedings of the First Conference of the Pacific Association for Computational Linguistics (PACLING '93),* Vancouver BC, Canada, 160-168.

Callmeier, U. 2000. PET – a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering* 6 (1): 99-108.

Carpenter, B. and G. Penn. 1999. *ALE: the Attribute Logic Engine. User's Guide. Version 3.2. Beta*.

Conway, A. and T. Veale. 1994. A linguistic approach to sign language synthesis. In: *People and Computers,* 211-222.

Copestake, A. 1999. The (new) LKB system. Stanford CA: CSLI. http://www-csli.stanford.edu/~aac/doc5-2.pdf

Copestake, A., D. Flickinger, R. Malouf, S. Riehemann and I. Sag. 1995. Translation using Minimal Recursion Semantics. In: *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*. Leuven, 15-32.

Copestake, A. and D. Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In: Gavrilidou, M. et al. (eds.), *Second International Conference on Language Resources and Evaluation: Proceedings*. Athens, 591-598.

Cormier, K., S. Wechsler and R.P. Meier. 1999. Locus agreement in American Sign Language. In: Webelhuth, G., J.-P. Koenig and A. Kathol (eds.), *Lexical and constructional aspects of linguistic explanation*. Chicago IL: Univ. of Chicago Pr., 215-229.

Ebbinghaus, H. and J. Heßmann. (in prep.) Sign language as multidimensional communication: why manual signs, mouthings, and mouth gestures are three different things. In: Boyes Braem, P. and R. Sutton-Spence (eds.), *The hands are the head of the mouth*. Hamburg: Signum.

Engberg-Pedersen, E. 1993. *Space in Danish Sign Language: the semantcis and morphosyntax of the use of space in a visual language*. Hamburg: Signum.

Erlenkamp, S. 1999. Possessivkonstruktionen in der Deutschen Gebärdensprache – warum die Gebärde SCH kein Verb ist. In: *Das Zeichen* 48, 274-279.

Habel, C. 1986. *Prinzipien der Referentialität: Untersuchungen zur propositionalen Repräsentation von Wissen*. Berlin: Springer.

Hanke, T. and S. Prillwitz. 1995. syncWRITER: integrating video into the transcription and analysis of sign language. In: Bos, H.F. and G.M. Schermer (eds.), *Sign Language Research 1994: Proceedings of the Fourth European Congress on Sign Language Research, Munich, September 1-3, 1994*. Hamburg: Signum, 303-312.

Johnston, T. and A. Schembri. 1999. On defining lexeme in a signed language. In: *Sign language and linguistics* 2 (2): 115-185.

Kamp, H. and U. Reyle. 1993. *From discourse to logic: introduction to model theoretic semantics of natural language, formal logic and Discourse Representation Theory*. Dordrecht: Kluwer Academic Publishers.

Levelt, W. J. 1989. *Speaking: from intention to articulation*. Cambridge MA: MIT Pr.

Liddell, S.K. 1995. Real, surrogate, and token space: grammatical consequences in ASL. In: Emmorey, K. and J.S. Reilly (eds.), *Language, gesture, and space*. Hillsdale NJ: Erlbaum, 19-41.

Marshall, I. and E. Safar. Extraction of semantic representations from syntactic CMU link grammar linkages. *Proceedings of Recent Advances in Natural Language Processing,* Bulgaria, 2001.

Meier, R. P. 1990. *Person deixis in American Sign Language*. In: Fisher, S.D. and P. Siple (eds.), *Theoretical issues in sign language research. Vol. 1: Linguistics*. Chicago IL/London: Univ. of Chicago Pr., 175-190.

Miller, G.A., R. Beckwith, Ch. Fellbaum, D. Gross and K. Miller. 1993. *An introduction to WordNet: an on-line lexical database*. http://www.cogsci.princton.edu/~wn/

Nerbonne, J., K. Netter and C. Pollard (eds.). 1994. *German grammar in HPSG*. Stanford CA: CSLI.

Pizzuto, E. and P. Pietrandrea. (in press). The notation of signed texts: open questions and indications for further research. In: *Sign Language and Linguistics* 3 (2).

Pollard, C., I.A. Sag. 1994. *Head-driven phrase structure grammar*. Stanford CA: CSLI.

Safar, E. and I. Marshall. 2001. Translation of English text to a DRS-based, sign language oriented semantic representation. *Proceedings of TALN Machine Translation Workshop*, Tours, July 2001.

Sag, I.A. and Wasow, T. 1999. *Syntactic theory: a formal introduction*. Stanford CA: CSLI.

Schembri, A. 2000. *Rethinking "classifiers" in signed languages*. Paper presented at the 7th International Conference on Theoretical Issues in Sign Language Research, Amsterdam, July 2000. (Unpubl. ms.; partial publication: Nachdenken über Klassifikatoren in Gebärdensprachen (Teil I). In: *Das Zeichen* 53, 470-483.)

Shieber, S.M., G. Van Noord, Moore, R.C. and F.C.N. Pereira. 1990. Semantic-head-driven generation. In: *Computational Linguistics* 16 (1): 30-42.

Sleator, D. and D. Temperley. 1991. *Parsing English with a Link Grammar. Carnegie Mellon University Computer Science technical report CMU-CS-91-196, October 1991.*

van Noord, G. 1989. *HDRUG: a graphical user environment for natural language processing in Prolog, Alfa-informatica, BCN*. University of Groningen, http://odur.let.rug.nl/~vannoord/Hdrug/

Uszkoreit, H., T. Becker, R. Backofen, J. Calder, J. Capstick, L. Dini, G. Erbach, D. Estival, S. Manandhar, A.-M. Mineur, G. van Noord and S. Oepen. 1996. *The EAGLES Formalisms Working Group. Final Report. Technical Report*. Saarbrücken: Deutsches Forschungszentrum für Künstliche Intelligenz.

Veale, T., A. Conway and B. Collins. 1998. The challenges of cross-modal translation: English-to-Sign-Language translation in the Zardoz system. In: *Machine Translation* 13: 81-106.

Wahlster, W. (ed.). 2000. *Verbmobil: foundations of speech-to-speech translation*. Berlin: Springer.

12 Appendix - Examples

The following examples are derived from batch executions of the system (rather than the interactive mode of the accompanying demonstrator). The output is left as commented loadable Prolog predicates, rules and assertions which are generated at intermediate stages. For each of four sentences, the original input, the DRS, the HPSG SEM, and the ALE synthesised BSL HamNoSys is shown.

12.1 The glass is broken.

INPUT

```
The glass is broken.
```

DRS generation

```
:-assert(drsEG(1,
"The glass is broken.",          % predicative interpretation
drs([v(0), v(1)],
[
  a(0):glass(v(0)),
  attr(0):broken(v(1)),
  t(0):when(s(0)),
  t(0)=now,
  s(0):be(v(0), v(1))
]

))).
```

DRS to HPSG SEM conversion

```
/*
The glass is broken.
*/
/*
sent ,                                  % pretty printed
sem :
mode : decl ,
index : SENT ,
restr : [
     (
     sit : SIT ,
     reln : be ,
     thm : nil ,
     location : nil ,
     figure : nil ,
     addressee : nil ,
     ground : nil ,
     instance : nil ,
     quant : nil ,
     act : ref , Indv1 ,
     arg : ref , Indv0 ,
     args : [
          (
          index : ref , Indv0 ,
          count : _G603 ,
          restr : [
               (
               sit : SIT ,
               reln : glass ,
               args : []
               )
               ]          ),

          (
          index : ref , Indv1 ,
          restr : [
```

```
            (
            sit : SIT ,
            reln : broken ,
            args : [
                 (
                 index : Indv1 , _G742
                 )
                 ]
            )
            ]
       )
       ]
  )
  ]
*/
/*---------------------------------------------*/
n1 :- gen((sent, sem: (mode:decl, index:SENT, restr:[ (sit:SIT, reln:be, thm:nil,
location:nil, figure:nil, addressee:nil, ground:nil, instance:nil, quant:nil, act:
(ref, Indv1), arg: (ref, Indv0), args:[ (index: (ref, Indv0), count:_G603, restr:[
(sit:SIT, reln:glass, args:[])]), (index: (ref, Indv1), restr:[ (sit:SIT,
reln:broken, args:[ (index:Indv1, _G742)])])])])),PPP),formatHamNoSysList(PPP,QQQ),
writeList(QQQ), writeListFile("hamNoSysS1to8",QQQ).
/*---------------------------------------------*/
```

SYNTHESISED HAMNOSYS (gloss [HamNoSys description (furrowed/raised/non_raised EYEBROWS - MANUAL HamNoSys)])

```
glass
    [ non_raised hamparbegin hamceeall hamplus hamflathand
      hamparend hamextfingerol hamparbegin hampalml hamplus
      hampalmu hamparend hampalm hamtouch hammoveu]
be
    []
broken
    [ non_raised hamsymmlr hamfist hamthumbacrossmod hamextfingeruo
      hampalmd hamtouch hamparbegin hammovedr hamarcu
      hamreplace hampalmul hamparend]
```

SIGML

```
<sigml>
  <hamgestural_sign gloss="glass">
    <sign_manual>
      <split_handconfig>
        <handconfig handshape="ceeall"/>
        <handconfig handshape="flat"/>
      </split_handconfig>
      <handconfig extfidir="ol"/>
      <split_handconfig>
        <handconfig palmor="l"/>
        <handconfig palmor="u"/>
      </split_handconfig>
      <location_hand location="palm" contact="touch"/>
      <directedmotion direction="u"/>
    </sign_manual>
  </hamgestural_sign>
  <hamgestural_sign gloss="be">
    <sign_manual/>
  </hamgestural_sign>
  <hamgestural_sign gloss="broken">
    <sign_manual both_hands="true" lr_symm="true">
      <handconfig handshape="fist" thumbpos="across"/>
      <handconfig extfidir="uo"/>
      <handconfig palmor="d"/>
      <handconstellation contact="touch"/>
      <par_motion>
        <directedmotion direction="dr" curve="u"/>
        <tgt_motion>
          <changeposture/>
          <handconfig palmor="ul"/>
```

```
      </tgt_motion>
    </par_motion>
  </sign_manual>
</hamgestural_sign>
```

12.2 Is this cup big?

INPUT

```
Is this cup big?
```

DRS generation

```
:-assert(drsEG(3,
"Is this cup big?",
drs([v(4), v(5)],
[
  attr(4):this(v(4)),
  a(2):cup(v(4)),
  attr(5):big(v(5)),
  qu(0):yesno(s(2)),
  t(2):when(s(2)),
  t(2)=now,
  s(2):be(v(4), v(5))
]

))).
```

DRS to HPSG SEM conversion

```
/*
Is this cup big?
*/
/*
sent ,
sem :
mode : yesnoqu ,
index : SENT ,
restr : [
     (
     sit : SIT ,
     reln : be ,
     thm : nil ,
     location : nil ,
     figure : nil ,
     addressee : nil ,
     ground : nil ,
     instance : nil ,
     quant : nil ,
     act : ref , Indv5 ,
     arg : ref , Indv4 ,
     args : [
          (
          index : ref , Indv4 ,
          restr : [
               (
               sit : SIT ,
               reln : this ,
               args : [
                    (
                    index : Indv4 ,
                    restr : [
                         (
                         sit : SIT ,
                         reln : cup ,
                         args : []
                         )
                         ]
                    )
                    ]
               )
```

```
              ]          ),

          (
          index : ref , Indv5 ,
          restr : [
               (
               sit : SIT ,
               reln : big ,
               args : [
                    (
                    index : Indv5 , _G6678
                    )
                    ]
               )
               ]
          )
          ]
     )
     ]
*/
/*--------------------------------------------*/
n3 :- gen((sent, sem: (mode:yesnoqu, index:SENT, restr:[ (sit:SIT, reln:be,
thm:nil, location:nil, figure:nil, addressee:nil, ground:nil, instance:nil,
quant:nil, act: (ref, Indv5), arg: (ref, Indv4), args:[ (index: (ref, Indv4),
restr:[ (sit:SIT, reln:this, args:[ (index:Indv4, restr:[ (sit:SIT, reln:cup,
args:[])])])]), (index: (ref, Indv5), restr:[ (sit:SIT, reln:big, args:[
(index:Indv5, _G6678)])])])]))),PPP),formatHamNoSysList(PPP,QQQ),
writeList(QQQ),writeListFile("hamNoSysS1to8",QQQ).
/*--------------------------------------------*/
```

SYNTHESISED HAMNOSYS (gloss [HamNoSys description
(furrowed/raised/non_raised EYEBROWS -
MANUAL HamNoSys)])

```
cup
    [ raised hampinch12open hamextfingero hampalml hamparbegin
      hammoveui hamarcu hamreplace hamextfingerul hamparend]
this
    [ raised hamfinger2 hamthumbacrossmod hampalml]
be
    []
large
    [ raised hamsymmlr hamceeall humthumbopenmod hamextfingero
      hampalml hamchest hamlrat]
```

SIGML

```
<hamgestural_sign gloss="cup">
    <sign_manual>
      <handconfig handshape="pinch12open"/>
      <handconfig extfidir="o"/>
      <handconfig palmor="l"/>
      <par_motion>
        <directedmotion direction="ui" curve="u"/>
        <tgt_motion>
          <changeposture/>
          <handconfig extfidir="ul"/>
        </tgt_motion>
      </par_motion>
    </sign_manual>
  </hamgestural_sign>
  <hamgestural_sign gloss="this">
    <sign_manual>
      <handconfig handshape="finger2" thumbpos="across"/>
      <handconfig extfidir="i"/>
      <handconfig palmor="l"/>
    </sign_manual>
  </hamgestural_sign>
  <hamgestural_sign gloss="be">
    <sign_manual/>
```

```
  </hamgestural_sign>
  <hamgestural_sign gloss="large">
    <sign_manual both_hands="true" lr_symm="true">
      <handconfig handshape="ceeall" ceeopening="slack"/>
      <handconfig extfidir="o"/>
      <handconfig palmor="l"/>
      <location_bodyarm location="chest" side="right_at"/>
    </sign_manual>
  </hamgestural_sign>
```

12.3 Where are the soup bowls?

INPUT

```
Where are the soup_bowls?
```

DRS generation

```
:-assert(drsEG(5,
"Where are the soup_bowls? ",
drs([v(8)],
[
  a(4):soup_bowl(v(8)),
  c(0)=count(v(8)),
  c(0)>1,
  t(4):when(s(4)),
  t(4)=now,
  s(4):be(v(8)),
  qu(2):where(s(4))
]

))).
```

DRS to HPSG SEM conversion

```
/*
Where are the soup_bowls?
*/
/*
sent ,
sem :
mode : whqu ,
index : SENT ,
restr : [
     (
     sit : SIT ,
     reln : where ,
     args : [
          (
          index : ref , Inds4 ,
          restr : [
               (
               sit : SIT ,
               reln : be ,
               thm : nil ,
               location : nil ,
               figure : nil ,
               ground : nil ,
               instance : nil ,
               addressee : nil ,
               quant : nil ,
               act : nil ,
               arg : ref , Indv8 ,
               args : [
                    (
                    index : ref , Indv8 ,
                    count : pl ,
                    restr : [
                         (
                         sit : SIT ,
                         reln : soup_bowl ,
```

```
                    args : []
                    )
                    ]
               )
               ]
          )
          ]
     )
     ]
)
]
*/
/*------------------------------------------*/
n5 :- gen((sent, sem: (mode:whqu, index:SENT, restr:[ (sit:SIT, reln:where, args:[
(index: (ref, Inds4), restr:[ (sit:SIT, reln:be, thm:nil, location:nil, figure:nil,
ground:nil, instance:nil, addressee:nil, quant:nil, act:nil, arg: (ref, Indv8),
args:[ (index: (ref, Indv8), count:pl, restr:[ (sit:SIT, reln:soup_bowl,
args:[])])])])])])])),PPP),formatHamNoSysList(PPP,QQQ),writeList(QQQ),writeListFile("
hamNoSysS1to8",QQQ).
/*------------------------------------------*/
```

SYNTHESISED HAMNOSYS (gloss [HamNoSys description (furrowed/raised/non_raised EYEBROWS - MANUAL HamNoSys)])

```
soup_bowl
    [ furrowed hamfinger2 hamfingerhookmod hamextfingerol hampalml
      hamshouldertop hammoveu hamarcd hamlips hamclose
      hamrepeatfromstartseveral hamplus hamsymmlr hamflathand hamfingerbendmod
      hamextfingeror hampalmul hamtouch hamparbegin hammoveur
      hamarcd hamreplace hampalml hamparend hamrepeatfromstart]
be
    []
where
    [ furrowed hamsymmlr hamfinger2345 hamthumboutmod hamextfingero
      hampalmu hamclose hamcircleu hamsmallmod hamrepeatfromstartseveral]
/*------------------------------------------*/
```

SiGML

```
<hamgestural_sign gloss="soup_bowl">
    <sign_manual>
      <sign_manual>
        <handconfig handshape="finger2" mainbend="hooked"/>
        <handconfig extfidir="ol"/>
        <handconfig palmor="l"/>
        <location_bodyarm location="shouldertop"/>
        <rpt_motion repetition="fromstart_several">
          <tgt_motion>
            <directedmotion direction="u" curve="d"/>
            <location_bodyarm location="lips" contact="close"/>
          </tgt_motion>
        </rpt_motion>
      </sign_manual>
      <sign_manual both_hands="true" lr_symm="true">
        <handconfig handshape="flat" mainbend="round"/>
        <handconfig extfidir="or"/>
        <handconfig palmor="ul"/>
        <handconstellation contact="touch"/>
        <rpt_motion repetition="fromstart">
          <par_motion>
            <directedmotion direction="ur" curve="d"/>
            <tgt_motion>
              <changeposture/>
              <handconfig palmor="l"/>
            </tgt_motion>
          </par_motion>
        </rpt_motion>
      </sign_manual>
    </sign_manual>
```

```
  </hamgestural_sign>
  <hamgestural_sign gloss="be">
    <sign_manual/>
  </hamgestural_sign>
  <hamgestural_sign gloss="where">
    <sign_manual both_hands="true" lr_symm="true">
      <handconfig handshape="finger2345" thumbpos="out"/>
      <handconfig extfidir="o"/>
      <handconfig palmor="u"/>
      <handconstellation contact="close"/>
      <rpt_motion repetition="fromstart_several">
        <circularmotion axis="u" size="small"/>
      </rpt_motion>
    </sign_manual>
  </hamgestural_sign>
```

12.4 There are knives in the sink.

INPUT

There are knives in the sink.

DRS generation

```
:-assert(drsEG(8,
"There are knives in the sink.",
drs([v(4), v(5)],
[
  a(4):sink(v(5)),
  l(2):in(v(4), v(5)),  % needs to lose 1 arg
  a(5):knife(v(4)),
  c(2)=count(v(4)),
  c(2)>1,
  t(2):when(s(2)),
  t(2)=now,
  s(2):exist(v(4))    % exist(v(4),l(2)) easier ?
]

))).
```

DRS to HPSG SEM conversion

```
/*
There are knives in the sink.
*/
/*
sent ,
sem :
mode : decl ,
index : SENT ,
restr : [
     (
     sit : SIT ,
     reln : exist ,
     location : ref , Indl2 ,
     arg : ref , Indv4 ,
     thm : nil ,
     figure : nil ,
     ground : nil ,
     addressee : nil ,
     instance : nil ,
     quant : nil ,
     act : nil ,
     args : [
          (
          index : ref , Indv4 ,
          count : pl ,
          restr : [
               (
               sit : SIT ,
               reln : knife ,
```

```
                args : []
                )
                ]          ),

            (
            index : ref , Indl2 ,
            restr : [
                (
                sit : SIT ,
                ground : ref , Indv5 ,
                location : nil ,
                arg : nil ,
                thm : nil ,
                figure : nil ,
                instance : nil ,
                quant : nil ,
                addressee : nil ,
                act : nil ,
                reln : in ,
                args : [
                    (
                    index : ref , Indv5 ,
                    count : sg ,
                    restr : [
                        (
                        sit : SIT ,
                        reln : sink ,
                        args : []
                        )
                        ]
                    )
                    ]
                )
                ]
            )
            ]
        )
        ]
*/
/*------------------------------------------*/
n8 :- gen((sent, sem: (mode:decl, index:SENT, restr:[ (sit:SIT, reln:exist,
location: (ref, Indl2), arg: (ref, Indv4), thm:nil, figure:nil, ground:nil,
addressee:nil, instance:nil, quant:nil, act:nil, args:[ (index: (ref, Indv4),
count:pl, restr:[ (sit:SIT, reln:knife, args:[])]), (index: (ref, Indl2), restr:[
(sit:SIT, ground: (ref, Indv5), location:nil, arg:nil, thm:nil, figure:nil,
instance:nil, quant:nil, addressee:nil, act:nil, reln:in, args:[ (index: (ref,
Indv5), count:sg, restr:[ (sit:SIT, reln:sink, args:[])])])])])])])),PPP),
formatHamNoSysList(PPP,QQQ),writeList(QQQ),writeListFile("hamNoSysS1to8",QQQ).
/*------------------------------------------*/
```

**SYNTHESISED HAMNOSYS (gloss   [HamNoSys description
                                   (furrowed/raised/non_raised EYEBROWS -
                                    MANUAL HamNoSys)])**

```
in
    []
sink
    [ non_raised hamsymmlr hamflathand hamextfingero hampalmu
      hamtouch hamparbegin hammoveur hamarcd hamreplace
      hampalml hamparend]
exist_have
    [ non_raised hamfinger2345 hamthumboutmod hamextfingeruo hampalmu
      hamparbegin hammoved hamsmallmod hamreplace hamfist
      hamthumbacrossmod hamextfingero hamparend]
knife
    [ non_raised hamsymmlr hamfinger23 hamthumbacrossmod hamextfingerol
      hampalml hamparbegin hamfingermidjoint hamindexfinger hamplus
      hamfingermidjoint hammiddlefinger hamparend hamtouch hamparbegin
      hammoveol hamsmallmod hamrepeatfromstart hamplus hamnomotion
      hamparend]
```

SiGML

```
<hamgestural_sign gloss="in">
    <sign_manual/>
  </hamgestural_sign>
  <hamgestural_sign gloss="sink">
    <sign_manual both_hands="true" lr_symm="true">
      <handconfig handshape="flat"/>
      <handconfig extfidir="o"/>
      <handconfig palmor="u"/>
      <handconstellation contact="touch"/>
      <par_motion>
        <directedmotion direction="ur" curve="d"/>
        <tgt_motion>
          <changeposture/>
          <handconfig palmor="l"/>
        </tgt_motion>
      </par_motion>
    </sign_manual>
  </hamgestural_sign>
  <hamgestural_sign gloss="exist_have">
    <sign_manual>
      <handconfig handshape="finger2345" thumbpos="out"/>
      <handconfig extfidir="uo"/>
      <handconfig palmor="u"/>
      <par_motion>
        <directedmotion direction="d" size="small"/>
        <tgt_motion>
          <changeposture/>
          <handconfig handshape="fist" thumbpos="across"/>
          <handconfig extfidir="o"/>
        </tgt_motion>
      </par_motion>
    </sign_manual>
  </hamgestural_sign>
  <hamgestural_sign gloss="knife">
    <sign_manual both_hands="true" lr_symm="true">
      <handconfig handshape="finger23" thumbpos="across"/>
      <handconfig extfidir="ol"/>
      <handconfig palmor="l"/>
      <handconstellation contact="touch">
        <location_hand location="midjoint" digits="2"/>
        <location_hand location="midjoint" digits="3"/>
      </handconstellation>
      <split_motion>
        <rpt_motion repetition="fromstart">
          <directedmotion direction="ol" size="small"/>
        </rpt_motion>
        <nomotion/>
      </split_motion>
    </sign_manual>
  </hamgestural_sign>
```