

ViSiCAST Deliverable D5-4: Integrated Editing Environment

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

Deliverable Number:	D5-4
Contractual Date of Delivery:	October 2002
Actual Date of Delivery:	December 2002
Title of Deliverable:	Integrated Editing Environment
Work-Package contributing to the Deliverable:	Workpackage 5 (Language and Notation)
Nature of the Deliverable:	PR (Prototype)
Author(s):	Ian Marshall (ed.) Thomas Hanke, Eva Safar, Constanze Schmaling, Hortensia Popescu, Sung-Eun Hong, Rachel Sutton-Spence

Abstract

Deliverable D5-4 is a prototype English text to Sign Language editing environment which allows input of English text from a file or from the keyboard to be converted into a Discourse Representation Structure (DRS) semantic representation from which a synthetic sign sequence is generated. User intervention by a linguistically aware person is supported to permit modification of the input text and modification of the synthesised sign language phonetic representation which drives the avatar. Human intervention is also permitted to choose between syntactic parse structures and discriminate between collective and distributive plurals. In addition, the prototype illustrates the use of anaphoric resolution within English to support use of placement within a model of signing space within the sign language grammar.

This deliverable embodies the first ever formulations of sign language grammars with linguistically interesting constructions and in sufficient detail to support synthetic generation of a phonological sign language representation and visualisation by virtual human (avatar) technology (developed in workpackage 4).

1	INTRODUCTION.....	3
2	BRIDGET JONES' DIARY	4
3	ARCHITECTURE.....	5
4	NLP ENGLISH TEXT PROCESSING – EXTENDED DRS HANDLING.....	5
5	NLP ENGLISH/SL DRS-HPSGSEM CONVERTER.....	10
5.1	DRS-DRS CONVERSIONS.....	10
5.2	DRS-HPSGSEM CONVERSION.....	11
6	SIGN LANGUAGE SYNTHESIS.....	12
6.1	LEXICON	12
6.1.1	<i>DGS lexicon size</i>	<i>12</i>
6.1.2	<i>Alphabet and spelling</i>	<i>13</i>
6.1.3	<i>Numbers and number incorporation</i>	<i>16</i>
6.1.4	<i>Plurals in ALE.....</i>	<i>19</i>
6.1.5	<i>Temporal Adverbs</i>	<i>20</i>
6.2	GRAMMAR	21
6.2.1	<i>Question pronouns and gap threading.....</i>	<i>21</i>
6.2.2	<i>Extended verb model.....</i>	<i>25</i>
6.2.3	<i>Signing space planning.....</i>	<i>30</i>
6.2.4	<i>Further grammatical phenomena implemented in ALE</i>	<i>34</i>
7	ACHIEVEMENTS AND LIMITATIONS.....	35
8	REFERENCES.....	36
9	APPENDICES	38
9.1	APPENDIX 1: DRS BNF / ONTOLOGY	38
9.2	APPENDIX 2: ACTUAL BJ TEXT INPUT.....	41
9.3	APPENDIX 3: BJ TEXT DRSS GENERATED	41
9.4	APPENDIX 4: BJ BSL HAMNoSYS GENERATED.....	43

1 Introduction

Deliverable 5.4 is an implementation of a semi-automatic text to sign preparation system, whose original functionality was characterised as inclusion of:-

- i. HamNoSys lexicons to cover a testable domain of about 1,000 concepts (including classifier verbs).
- ii. a user interface for a semi-automatic English to Sign Language Notation translation tool supporting manual intervention and previewing of signed sequences.
- iii. an SL formulator/planner to model the signing space (to support the introduction of new discourse referents and anaphora-like references to these).
- iv. an extension of the English to DRS translation system [of Deliverable D5-3] to handle the tense system and temporal and locative complement phrases.

As the ViSiCAST project progressed this description was refined to allow variation in number of signed concepts for different sign languages. For German Sign Language (DGS) a large corpus of HamNoSys transcribed sequences has supported construction of a large lexicon of 3,600 signs (See Section 6.1.1 below). In the case of the British Sign Language (BSL) the lack of such a resource has resulted in a lexicon limited to approximately 250 signs, though displaying a comparable variety in the type of signs.

In addition, the lexicons have been focused towards signing of a narrative describing a scene from the film 'Bridget Jones' Diary', in order to demonstrate the capabilities of the prototype and to generate a sign sequence which demonstrates coherence over multiple propositions. The choice of film, and scene from within it, was motivated by an initial concentration on a 'kitchen world' domain which informed the early stages of lexicon development. Hence, the kitchen scene from this film presented a natural progression of this work within a context comparable with signing presentations from a text stream source (such as subtitles). A methodology for data capture in terms of relevant syntactic phenomena and lexicon derivation was devised to assist this process. Deaf informants for each of the sign languages were asked to view the scene from the film (repeatedly if necessary) and then describe the scene to camera. These commentaries were video recorded and then analysed for their lexical (sign) content and the grammatical features they exhibit. Lexicon and grammar development were then informed by this data. The synthesised sign sequence is illustrative of descriptive narratives that a text to signing system may be expected to support.

The integration of the components of the English to Sign Language phonological translation and the signing avatar is detailed in Deliverable D5-3 and Milestone M5-11. This report summarises these but concentrates on the Natural Language Processing work since Deliverable D5-3. The foundations for this deliverable were laid in deliverables D5-1 and D5-3 which have proven robust and are carried forward in the main into this deliverable. Section 2 describes the choice of domain as the basis for the lexicon and grammar development as well as the methodology of the data collection. Section 3 reviews briefly the overall architecture and illustrates the demonstrable system. Section 4 describes the modifications and extensions to the English text analysis part of the system. Section 5 describes the conversion between the two central semantic notations, the English oriented DRS and the sign language oriented HPSGsem representations. Section 6 details the extensions to the sign language grammars and lexicons developed during the past twelve months. Section 7 critically reviews the successes and limitations of this research.

The DGS HPSG synthesis system is implemented in LINGO and the BSL in ALE. Therefore we refer to the two different sub-systems by the language or by the implementation throughout the text.

2 Bridget Jones' Diary

The choice of film, and scene from within it, was motivated by an initial concentration on a 'kitchen world' domain which informed the early stages of the lexicon development. Hence, the kitchen scene from this film presented a natural progression of this work within a context comparable with signing presentations from a text stream source (such as subtitles). A methodology for data capture in terms of relevant syntactic phenomena and lexicon derivation was devised to assist this process.

Three deaf native signers, two for DGS and one for BSL, were asked to view the scene from the film (repeatedly if necessary) and then describe it to camera. The DGS narratives lasted 6:45 minutes and 9:17 minutes respectively. The signed stories were transcribed in detail with glosses, using syncWRITER (cf. Hanke and Prillwitz 1995). In addition to the glosses, mouthing, eye gaze, and non-manual features, as well as manner of articulation were transcribed. It was also possible to add information about classifier handshapes, and role-shift (see Figure 1).










									
	0:00:13.52	0:00:14.44	0:00:16.28	0:00:17.60	0:00:18.44	0:00:19.44	0:00:20.16	0:00:20.76	0:00:21.60
Glossen	NBRIDGET		SPAZIEREN	MARKT	SPAZIEREN	DENKEN	LIED	SINGEN	DE
nonmanualHand									
mouthing			m	m					
ArtundWeise									
Nonmanualia	Kopf sackt mit nach vorne		Kopf geht mit		Kopf geht mit				
Auge						nach oben		na	
allgemein	überlegt, setzt in ihrem gebärdenfluss kurz aus						role-shift Bridget		

Figure 1

Both the development of the lexicon and the grammar were driven by this data. From the data, a list of glosses was established for both languages and served as a basis for the entries in the HPSG lexicon. Using syncWRITER offered the possibility to go back to each sign and see how it is performed. However, if the exact production of a sign was not clear from the filmed narration, other native signers were also asked to perform these signs. Information on classifier handshapes – part of the HPSG entries – was based on the use of classifiers in the narrated sequences as well. In some cases, the use of classifiers was cross-checked with other native signers at the IDGS. Other native signers were also interviewed if the two signers differed on their use of certain signs, e.g. regarding some directional signs, or the use of AUF.

Finally, a back translation of each of the signed sequences into written English was prepared. These served as input for the generation of signed phrases of the avatar. The initial sentences of this text are given in Appendix 2 and output at subsequent processing stages are given in Appendices 3 (generated DRSSs) and 4 (generated HamNoSys). These are demonstrable with the implemented system.

3 Architecture

The architecture of Deliverable D5-4 is illustrated in Figure 2. English text is parsed by the Carnegie Mellon University link grammar parser and its output (a linkage) is passed to a component which extracts a Discourse Representation Structure (DRS) from this representing the logical content and relevant pragmatic features for each English sentence. The DRS is then converted to a Head-Driven Phrase Structure Grammar (HPSG) semantic form (HPSGsem) for each of the target sign languages. During this stage conversion of n-ary predicates derived from English may be realised as p-ary predicates for signing. Each sign language has its own HPSG grammar and lexicon with which sign sequence phonological descriptions are generated and passed to the avatar for visualisation.

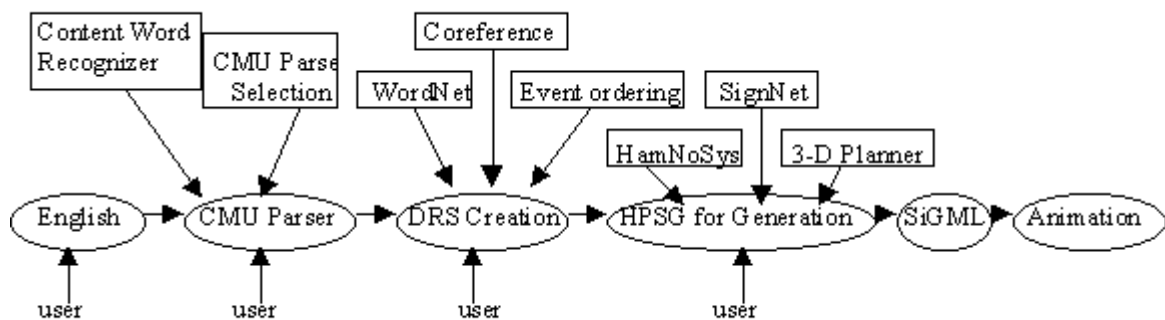


Figure 2

Figure 3 illustrates the realisations of these stages within the implemented system. Human interaction within these processes by a linguistically informed individual is supported to allow modification of the input text, indication of collective versus distributive plurals and to modification of the HamNoSYs gesture notation prior to visualisation within the avatar. In addition the system implementation allows multiple sentences to be treated as a discourse unit, resolving pronominal references thus ensuring that such references are associated with the same location in signing space. Furthermore, the generated DRS and status of the signing space planner is supported.

4 NLP English text processing – Extended DRS Handling

Deliverable D5-1 Section 2 details the Discourse Representation Structure (DRS) used as an intermediate semantic representation of the natural language processing of English text to Sign Language. The original specification was relatively comprehensive of SL phenomena to be considered and provision to be catered for in the DRS. This detail is not repeated here and the reader is referred to that document for the original formulation. Hence, this document details the main though relatively minor revision to that description during progress of the work.

The DRS semantic representation is generated from the linkage output from the CMU parser. This is now implemented for approximately 60% of link types but these account for the most common linguistic phenomena.

In addition to the revision to the DRS representation detailed below, the handling of DRS structures has been extended to support both inter- and intra-sentential pronominal reference.

In sign languages, pronouns are realised as pointing gestures to the location associated with a noun and as start and end positions of directional verbs. These languages make extensive use of placement of referents at particular points in the signing space, therefore anaphora resolution in English text is crucial for a correct translation into BSL.

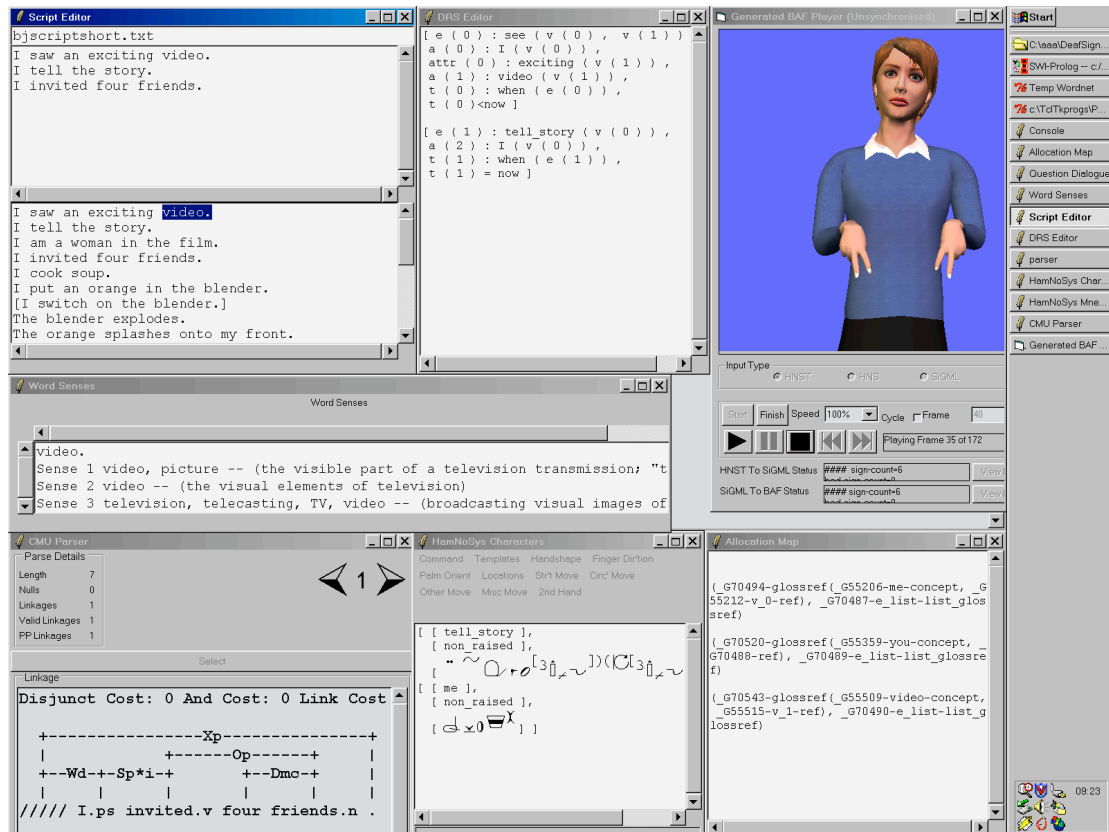


Figure 3

A significant number of antecedents which are potential referents for a pronoun in English can be excluded by linguistic restrictions on gender-number agreement, intra- and intersentential accessibility constraints in DRSs. These constraints are applied in a small window (two sentences), avoiding complex semantic and discourse analysis. This is vital for a translation system as a real-world application.

Antecedents that obey the above mentioned constraints are scored by preferences. The weighting algorithm is a modified version of the work by (Kennedy and Boguraev 1996). They claim "the strong points of this algorithm is that it operates primarily on syntactic information alone" with a 75% accuracy rate. The current implementation is an improvement to the gender agreement in (Kennedy and Boguraev 1996) by augmenting the algorithm with a lexical database (female, male names, WordNet), to conditions on coreferents in (Kennedy and Boguraev 1996) by making use of accessibility constraints in DRS. The modified algorithm improves on the suitability idea of (Kamp and Reyle 1993) by determining how to choose from more than one referent in the DRS.

The original Kennedy/Boguraev algorithm calculated the salience weight of a possible referent as the sum of the salience factors (grammatical role, adjunct, embedding, current sentence and complement of a preposition, existential construction). As each CMU link has an entry in the link dictionary defining its associated lambda-DRS expression, a salience value is associated directly in its link dictionary entry. Following (Kennedy and Boguraev 1996) a COREF class is the collection of linguistic elements that corefer in a text to describe the same discourse referent. A COREF salience is associated with each of these. When an anaphor has to be resolved, the

COREF class with the highest salience is selected. This possible referent is then checked for agreement in number, gender and accessibility within the DRS. Number agreement is checked with a noun stemming algorithm, the gender of nouns is looked up in a database with female and male proper names. Potential referents that do not satisfy these requirements are removed. When a link between a discourse referent (which can be another anaphor) and the current anaphor is established, this becomes a member of that class and its salience value is set to the COREF value of the antecedent. For each new sentence the COREF values of previous sentences are halved. Thus, the salience of a COREF class increases according to the frequency of subsequent anaphoric references to it and decreases otherwise.

The intrasentential pronoun resolution follows the Kamp and Reyle (1993) introduced in their work. If the sentence contains anaphoric pronouns, their antecedents must be found in the set of available and suitable referents (e.g.: x) in the universe (U). The construction rule then introduces a new discourse referent in the subordinate universe with the pronoun replaced in the sentence by a variable, for example w . The equation $w=x$ is added to the conditions. In our case the variables are immediately substituted by the resolved anaphora without adding the equation to the conditions. For further details see Kamp and Reyle (1993).

In deliverable D5-1 (section 2.2) plurals were given only a cursory treatment with the detail to be determined as work progressed. This has now been refined consistent with the treatment of plurals in Kamp and Reyle (1993), allowing the user to volunteer information to distinguish between collective and distributive plural forms. Note that Kamp and Reyle (1993) permit predicates over sets with the meaning that the predicate holds of each element of the set, hence `plate(set(0))` below.

These two forms are realised respectively as

- (i) sentence1 The large plates are on the small plates. (Collective of large plates.)
 (The set of large plates is on the set of small plates)

```
drs(0):[set(0), set(1)],
[
  set(0)=sumof(v(0)),
  t(0):when(s(0)),
  t(0)=now,
  s(0):be(set(0)),
  set(1)=sumof(v(1)),
  l(0):on(s(0), set(1)),
  attr(0):small(set(1)),
  a(0):plate(set(1)),
  c(0):plural(set(1)),
  attr(1):large(set(0)),
  a(1):plate(set(0)),
  c(1):plural(set(0))
]
```

- (ii) sentence2 The large plates are on the small plates. (Distributive of large plates.)

(Each large plate is on one or more small plate)

```
drs(1):[set(2)],
[
  set(2)=sumof(v(2)),
  drs(2):[v(2)],
  [
    q(0):every(v(2)),
    attr(2):large(v(2)),
    a(2):plate(v(2)),
    c(2):plural(v(2))
  ]
  >
  drs(3):[set(3)],
  [
    t(1):when(s(1)),
    t(1)=now,
    s(1):be(v(2)),
    set(3)=sumof(v(3)),
    l(1):on(s(1), set(3)),
    attr(3):small(set(3)),
    a(3):plate(set(3)),
    c(3):plural(set(3))
  ]
]
```

Phrases such as ‘every man’ are similarly permitted collective and distributive readings.

sentence3 Every man loves a woman in the park. (Collective reading.)

```
drs(0):[set(0), v(0), v(1)],
[
  set(0)=sumof(v(2)),
  a(0):woman(v(0)),
  t(0):when(e(0)),
  t(0)=now,
  e(0):love(set(0), v(0)),
  a(1):park(v(1)),
  l(0):in(e(0), v(1)),
  a(2):man(set(0))
]
```

sentence4 Every man loves a woman in the park. (Distributive reading.)

```
drs(0):[set(0)],
[
  set(0)=sumof(v(0)),
  drs(1):[v(0)],
  [
    q(0):every(v(0)),
    a(0):man(v(0))
  ]
  >
  drs(2):[v(1), v(2)],
  [
    a(1):woman(v(1)),
    t(0):when(e(0)),
    t(0)=now,
    e(0):love(v(0), v(1)),
    a(2):park(v(2)),
    l(0):in(e(0), v(2))
  ]
]
```

]

Furthermore, numerically quantified plurals receive a comparably consistent treatment.

sentence5 Three lawyers hired a secretary. (Collective reading.)

(A group of three lawyers hired a secretary.)

```
drs(0):[set(0), v(0)],
[
  set(0)=sumof(v(1)),
  size(set(0))=three,
  a(0):secretary(v(0)),
  t(0):when(e(0)),
  t(0)<now,
  e(0):hire(set(0), v(0)),
  a(1):lawyer(set(0)),
  c(0):plural(set(0))
]
```

sentence6 Three lawyers hired a secretary. (Distributive reading.)

(Each of three lawyers hired a secretary.)

```
drs(1):[set(1)],
[
  set(1)=sumof(v(2)),
  size(set(1))=three,
  drs(2):[v(2)],
  [
    q(0):every(v(2)),
    a(2):lawyer(v(2)),
    c(1):plural(v(2))
  ]
  >
  drs(3):[v(3)],
  [
    a(3):secretary(v(3)),
    t(1):when(e(1)),
    t(1)<now,
    e(1):hire(v(2), v(3))
  ]
]
```

The revisions to the BNF description of the DRS form to incorporate this are

SetDefinition :-

SetVar = sumof(Var)

CollectiveSet :-

Cvar : plural(SetVar)

DistributiveSet :-

Qvar : Quant(Var)

Cvar : plural(Var)

Quant :-

every | some | few | many

QuantifiedSet :-

size(SetVar) = NumericValue

NumericValue :-
 one | two | three | four | ...

which replace the previous *CollectiveDefinition*, *NumericalQuantifiedDefinition*, *QuantifiedVariable*, *UniversalQuantification* and *ExistentialQuantification* productions. The full revised BNF description is given in Appendix 1. Remaining queries within Appendix 1 did not give rise to issues during the timescale of ViSiCAST.

The distinction between the DRS structures is carried forward as a HPSG feature COLLORDIST with values of COLL(ective) or DISTRIB(utive) which allows the two forms to be distinguished in the HPSG SL synthesis stage.

5 NLP English/SL DRS-HPSGsem converter

At the centre of the text to sign translation system is a sub-component which transforms the DRS representation

- i. to a more sign language oriented DRS form in which predicate valencies are modified to conform with predicates supported in the sign language lexicon,
- ii. and then to a HPSG-sem(antic representation) from which HPSG synthesis commences.

As the DGS HPSG synthesis system is implemented in LINGO and uses a variation on the shake and bake generation algorithm, the required HPSGsem is a flat structure in which indexes denote coreferential structures. Conversely, the BSL/NGT synthesis system is implemented in ALE and use a Head Driven generation algorithm and necessitates a nested HPSG-sem input structure. After (i) above the two systems kinds of system diverge, each using a different version of the DRS-HPSGsem converter, though designed on similar principles.

5.1 DRS-DRS conversions

The DRS to DRS conversions are concerned with resolving differences in the valencies of state and event predicates which initially reflect their use in English with similarly named SL predicates of different valencies. For example,

- i English forms of the verb ‘put’ are analysed as involving two complements, the agent and the patient object, and an adjunct, the destination location (derived from a locative prepositional phrase). In the lexical entries for ‘put’ in the target sign languages these are three complement verbs. The English oriented DRS form has the predicates of the form

$$e(N) : \text{put}(v(X), v(Y)), l(Q) : \text{in}(e(N), v(Z))$$

which are transformed into the SL oriented form

$$e(N) : \text{put}(v(X), v(Y), v(Z)).$$

Similar transformations exist for other verbs of motion such as ‘take’ and ‘move’ in cases where the valency of the SL verb form is greater than the required English complements.

- ii For a number of sign language verbs, the manner or direction of the movement is incorporated within the sign. For example, ‘walk upstairs’ and ‘walk downstairs’ are realised by two different lexical items in which the adverbial component is incorporated within the sign. Hence for the former case, a transformation which replaces

$$e(N) : \text{walk}(v(X)), l(Q) : \text{upstairs}(e(N))$$

by

$e(N) : \text{walk_upstairs}(v(X))$

is used.

- iii For a number of sign language verbs, information regarding the patient object is incorporated within the sign. For example, ‘to put on an apron’ is signed distinctly differently from ‘to put on the kettle’. Hence for the former case, a transformation which replaces

$e(N) : \text{put_on}(v(X), v(Y)), a(Q) : \text{apron}(v(Y))$

by

$e(N) : \text{put_on_apron}(v(X), v(Y)), a(Q) : \text{apron}(v(Y))$

is used. In this case the valency of the SL verb predicate is the same as the English counterpart and the predicate regarding the patient is retained. The former determines the appropriate verb selection from the SL lexicon, whilst the latter permits later techniques which control placement within signing space to still be informed about the patient nominal.

It is conceivable that cases (ii) and (iii) in the longer term may be addressed by comparable SL generation mechanisms to those which allow generic characterisations of directional verbs to be instantiated with specific locations in signing space. Generalisation of this technique to less specific locations and alternative sources of handshape classifier information is an interesting longer term prospect, however in the interim these are handled by this source-target DRS transfer mechanism.

5.2 DRS-HPSGsem conversion

The HPSGsem generation is a top-down recursive process which actively seeks a main proposition (state or event predicate) and attributive propositions relating to its arguments. For example, the DRS

```
[ e ( 2 ) : invite ( v ( 0 ) , set ( 0 ) ) ,  
  a ( 7 ) : I ( v ( 0 ) ) ,  
  set ( 0 ) = sumof ( v ( 8 ) ) ,  
  size ( set ( 0 ) ) = four ,  
  t ( 3 ) : when ( e ( 2 ) ) ,  
  t ( 3 ) < now ,  
  a ( 8 ) : friend ( set ( 0 ) ) ,  
  c ( 0 ) : plural ( set ( 0 ) ) ]
```

is transformed into the corresponding ALE HPSG-sem.

```
sent,  
sem: (mode:decl, index:_G548620,  
  restr:[ (sit:_G548679, reln:invite, arg:nil, location:nil,  
    figure:nil, ground:nil, addressee:nil,  
    instance:nil, quant:nil, act: (ref, v_0), thm: (ref, set_0),  
    sense:_G548745,  
    args:[ (index: (ref, v_0), topic:comment,  
      count: (number:sg, collordist:_G548854),  
      restr:[ (sit:_G548866, reln:me, sense:_G548878,  
        args:[])]),  
      (index: (ref, set_0), topic:comment,  
        restr:[ (sit:_G549158, reln:four, sense:_G549170,  
          args:[ (index: (ref, set_0), topic:comment,  
            count: (number:pl, collordist:coll),  
            restr:[(sit:_G548971, reln:friend,  
              sense:_G548983, args:[])]
```


building and transcription, currently contains more than 6,000 signs. The available entries were reviewed and tagged as one of the following:

- already included in the HPSG lexicon
- fixed verbs belonging to certain valence classes
- locatable or fixed nouns referring to person, object, or location
- too complex to be automatically transferred

A Perl script was then used to create HPSG entries from the data (including the tags) of the suitable candidates, with the PHON being reconstructed from citation form HamNoSys data given in the database or manually revised.

Due to the aims of the data collection tasks for the lexical database, the number of entries sharing their semantic content with other entries is relatively high. In the case of the 3,000 signs transferred so far, the number of semantic concepts covered is about 1,800.

Work will be continued to explore possibilities to classify and semi-automatically transfer entries for the remaining 2,500 entries.

6.1.2 Alphabet and spelling

6.1.2.1 Lingo

6.1.2.1.1 Letters

For the letters of the alphabet, we introduced a new type, *alpha-t*:

```
alpha-t := non-compound-lxm &
[ SYN.HEAD nopos ].
```

Initially, the part-of-speech *pos* value of the letters was unspecified (the feature HEAD was of type *pos*) but in order to keep the letters from combining with other lexemes, we introduced a new subtype of *pos*, namely *nopos*.

Each *alpha-lxm* has the following feature structure:

```
alpha-lxm := alpha-t &
[ PHON.MAN spelling-struct,
  SYN [HEAD.AGR.COORD true,
      PRECOMPS < >,
      POSTCOMPS < >,
      QPRO false ],
  SEM.RESTR <! spelling-predication !> ].
```

The letters have no PRECOMPS and no POSTCOMPS. The feature SEM.RESTR specifies a restriction of type *spelling-predication*:

```
spelling-predication := predication &
[ ALPHA alpha,
  RELN spelling_r ].
```

In order to describe the PHON.MAN of the letters, we introduced *spelling-struct* as a new subtype of the type *man-struct*:

```
spelling-struct := man-struct &
[ MOV.MLOC <! hamshoulders,hamlrat !> ].
```

There are three alphabet letter types in DGS, single letter (*single-letter-spelling-struct*), double letter (*double-letter-spelling-struct*), and umlaut (*umlaut-spelling-struct*), which differ as to their respective PHON.MAN part. They are subtypes of the type *spelling-struct*, and are only specified for the features, ORI and MOV:

```
single-letter-spelling-struct := spelling-struct &
  [ ORI [EFD <! hamextfingeru !>,
    PLM <! hampalmd !> ],
    MOV.M-HNS <! hammoveo,hamsmallmod !>].

double-letter-spelling-struct := spelling-struct &
  [ ORI [EFD <! hamextfingeru !>,
    PLM <! hampalmd !> ],
    MOV.M-HNS <! hammover !> ].

umlaut-spelling-struct := spelling-struct &
  [ ORI [EFD <! hamextfingeru !>,
    PLM <! hampalmd !> ],
    MOV.M-HNS <! hammoved,hamsmallmod,hamrepeatfromstart !>].
```

Each letter gets its PHON.MAN from the *spelling-struct* and adds a specific handshape, e.g.:

```
alphaA_1 := alpha-lxm &
  [ GLOSS <! "A" !>,
    PHON.MAN single-letter-spelling-struct & [ HSH <! hamfist !> ],
    SEM.RESTR <! [ ALPHA alpha_A ] !> ].

alphaAA_1 := alpha-lxm &
  [ GLOSS <! "AA" !>,
    PHON.MAN double-letter-spelling-struct & [ HSH <! hamfist !> ],
    SEM.RESTR <! [ ALPHA alpha_AA ] !> ].
```

6.1.2.1.2 Combining letters to words

The lexical rule *spelling* builds a sequence of letters. It takes as arguments a sequence of letters and an individual letter. Because the type of this sequence is *alpha-t*, the sequence may also only consist of one letter only. The SYN value of the rule (the resulting sequence) is the same as that of an individual letter, and the PHON feature of the rule consists of the concatenation of the PHON values of the daughters.

```
spelling := lrule-alpha &
  [ GLOSS [ LIST #first,
    LAST #last ],
    PHON.SUBEVENTS < #sub1, #sub2 >,
    SYN #syn,
    SEM.RESTR [ LIST #firstsem,
    LAST #lastsem],
    PHON [],
    ARGS < alpha-t & [ SYN #syn,
    SEM.RESTR [ LIST #firstsem,
    LAST #middlesem ],
    GLOSS [LIST #first,
    LAST #middle ],
    PHON #sub1 ],
    alpha-lxm & [ SYN #syn,
    SEM.RESTR [ LIST #middlesem,
    LAST #lastsem ],
    GLOSS [LIST #middle,
    LAST #last ],
    PHON #sub2 ] > ].
```

The lexical rule *alpha-to-word* is a LEXEME to WORD rule (it transforms a *lexeme* into a *word*). The rule modifies the *pos* of a *lrule-alpha* (from *nopos* to *noun*): it requires the HEAD feature of the daughter to be of type *nopos* and its HEAD to be of type *noun*. This way the mother cannot be combined with other letters because both daughters of the *spelling* rule are required to be HEAD *nopos*.

```
alpha-to-word := lrule-word &
[ GLOSS #gloss,
  PHON #phon,
  SYN [HEAD noun & [ AGR.COORD true ],
      PRECOMPS < >,
      POSTCOMPS < >,
      GAP <! !>,
      QPRO false,
      REFPT-IN < >,
      REFPT-OUT < > ],
  SEM #sem,
  ARGS < lrule-alpha & [GLOSS #gloss,
                        PHON #phon,
                        SYN [ HEAD nopos ],
                        SEM #sem ] > ].
```

6.1.2.2 ALE

6.1.2.2.1 Letters

In the lexicon a new type has been introduced: *letter* to distinguish between *words* and *letters*. This type has the head (subtype) *alphabet*.

```
[ [ c ],
  [Brow ],[hamcee12, hamthumbopenmod, hamextfingeru, hampalml ]]
  --->
  letter,
    phon:(face:brow:Brow),
    syn:(precomps:[ ],
         postcomps:[sem:X],
         arg_st:[sem:X],
         head:(alphabet, context:(context_in : Cin,
                                   context_out: Cin))),
    sem:(
      restr:[(
        reln:c,
        args:[X])]).
```

Letters as lexical items have a non-specified postcomplement to allow the chaining of letters.

6.1.2.2.2 Rules

Fingerspelling is determined by three rules in ALE. The recursive rule at the bottom level is the *spelling rule*, whose head-daughter is a *letter* and its daughter is already a *word* with no further complements. The head-daughter still requires a complement but its mother *postcomp* feature is empty. This means that in a sequence it is always the next letter that governs the generation as the head. The daughter as partial word is attached to it. The *spelling_last rule* is used if no further letters remain in the generation. This rule connects bottom-up to the next rule, which is the *fingerspelling rule*. From the mother of this rule it is possible to connect back to other grammar rules' daughter or head-daughter with instantiated classifier and head (*noun*) information. However this rule's function as a non-chain rule to find the pivot first, which is then linked to the root bottom-up by the chain rules.

6.1.2.2.3 Principles

If an object's or person's name is fingerspelled, it is still vital that classifier information is propagated to the verb to instantiate the handshape feature value in the PHON and the LHS of the verb. The *spelling_cl* principle is used to determine the needed classifier information. Currently we only handle persons' names, therefore the principle is:

```
spelling_cl(person, (cl_manip: (@ one_handed_twoD_C),
                        cl_subst: (@ stick))) if true.
```

Here *person* stands for the value of the *gloss* as well. This is needed because of the allocation of signing space in non-recursive grammar rules (see also section 6.2.3).

6.1.3 Numbers and number incorporation

6.1.3.1 DGS

6.1.3.1.1 Numbers

All numbers are determiners of type *det-lxm*:

```
det-lxm := lexeme &
  [ SYN [HEAD det,
        PRECOMPS < >,
        POSTCOMPS < >,
        QPRO false ],
    SEM.MODE /1 null-mode ].

cardinal-lxm := det-lxm & non-compound-lxm &
  [ SEM.RESTR <! quantity-predication !> ].

cardinalnon1-lxm := cardinal-lxm &
  [ SYN.HEAD.AGR.NUM p1 ].
```

Except for the number 1 which modifies a singular referent, all other numbers (2–1000, 2000, 3000,...,10000) modify plural referents.

We differentiate between two different types of lexemes, for the numbers up to ten (*uptoten-lxm*) and for numbers above ten (*aboveten-lxm*), because the numbers 1–10 can be incorporated into a limited set of DGS signs whereas the numbers above 10 cannot.

```
uptoten-lxm := cardinalnon1-lxm &
  [ SYN.HEAD uptoten,
    PHON.MAN [ ORI [EFD <! hamextfingeru !>,
                  PLM <! hampalmu !> ],
              MOV [MLOC <! hamshoulders,hamlrat !>,
                  M-HNS <! hammoveo,hamsmallmod !> ]]]].

aboveten-lxm := cardinalnon1-lxm &
  [ SYN.HEAD aboveten ].
```

Aboveten-lxm has four subtypes, *teen-lxm*, *ties-lxm*, *hundred-lxm*, *thousand-lxm*.

These number lexemes are not specified for handshape but only for the features ORI and MOV, e.g.:

```
teen-lxm := aboveten-lxm &
  [ PHON.MAN [ ORI [EFD <! hamextfingero !>,
                  PLM <! hampalmd !> ],
              MOV.M-HNS <! hamcirclel,hamsmallmod,hamrepeatfromstart !> ]].
```

Each type has a subtype for those number signs produced with two hands; and they make use of either parallel or mirror symmetry, e.g.:

```
hundred2-lxm := hundred-lxm & two-handed-par.
```

```
teen2-lxm := teen-lxm & two-handed-lr.
```

6.1.3.1.2 Number incorporation

Each number is of one of the above-mentioned lexeme types. In the PHON.MAN part, a number handshape is incorporated from the *numbers* type.

```
two_1 := uptoten-lxm &
[ GLOSS <! "TWO" !>,
  PHON.MOUTH [ PICT "tsvaI",
               PICT-DE "zwei" ],
  PHON.MAN number_2,
  SEM.RESTR <! [QUANT nr_2] !> ].
```

For these handshapes, we introduced a new subtype of *man-struct*, called *numbers* which has two subtypes, *one-handed-number* and *two-handed-number*:

```
numbers <: man-struct.
```

```
one-handed-number := numbers & one-handed.
```

```
two-handed-number := numbers & two-handed &
[ CONST emptyhamnosys,
  WEAK-HAND cant-drop ].
```

Each number handshape is of one of these two subtypes, e.g.:

```
number_2 := one-handed-number &
[ HSH <! hamfinger23spread,hamthumbacrossmod !> ].
```

Numbers above 20, except the *-ties*, *-hundreds*, and *-thousands*, are all compound lexemes whose manual PHON part is composed of two subevents which occur sequentially:

```
cardinalnon1-compound-lxm := aboveten-lxm &
[ PHON.SUBEVENTS < [ MAN #1 ], [ MAN #2 ] >,
  ARGS < [ PHON.MAN #1 ], [ PHON.MAN #2 ] > ].
```

Example entry:

```
ninety-six_1 := cardinalnon1-compound-lxm &
[ GLOSS <! "NINETY-SIX" !>,
  PHON.MOUTH [ PICT "zEksUntnOYntsIC",
               PICT-DE "sechsendneunzig" ],
  ARGS < cardinalnon1-lxm & [GLOSS <! "NINETY" !>],
               uptoten-lxm & [GLOSS <! "SIX" !>]>,
  SEM.RESTR <! [QUANT nr_96] !> ].
```

Note that mouthing is specified at top level in order to avoid the necessity for a grammar of the spoken language number system.

We have seen how the handshapes for the numbers 1–10 are incorporated in the cardinal numbers (numbers above 10 cannot be incorporated). In a similar way, these handshapes are incorporated in ordinal numbers and in a group of signs we have called temporal adverbs. These signs include OCLOCK, HOUR, WEEK, MONTH, and YEAR. CENT is another sign where number handshapes can be incorporated. All of these signs can incorporate the numbers 1–10.

Incorporation works in the same way as described above for the cardinal numbers. The different types (*oclock-lxm*, *hour-lxm*, *week-lxm*, *month-lxm*, *year-lxm*) – all specified for ORI and MOV only – are all subtypes of the *tempadv-incorp-lxm*.

```
tempadv-incorp-lxm := tempadv-lxm &
  [ SYN [ PRECOMPS < >,
          POSTCOMPS < >,
          QPRO false ]].
```

A lexeme of type *week-lxm* gets its handshape from the *numbers* subtype of *man-struct*, e.g.:

```
week-lxm := tempadv-incorp-lxm &
  [ PHON.MAN [ ORI [ EFD <! hamextfingeru !>,
                   PLM <! hampalmu !> ],
             MOV [ MLOC <! hamshoulders !>,
                   M-HNS <! hamparbegin,hammover,hamreplace,hampalmd,hamparend !>
             ]]].
```

```
fourweeks_1 := week-lxm &
  [ GLOSS <! "FOURWEEKS" !>,
    PHON [ MOUTH [ PICT "fi:6vOx@n",
                   PICT-DE "vierwochen" ],
          MAN number_4 ],
    SEM.RESTR <! [TEMP temp_fourweeks] !> ].
```

6.1.3.2 BSL

In ALE numbers belong to the type of quantifier lexemes (*quantif_lxm*). In BSL they are pre-modifiers.

```
[[two],
 [Brow], ['tu:', hamfinger23spread, hamthumbacrossmod, hamextfingeru, hampalmu, hams
 houlders, hamlrat]]--->
  word,
  gloss:two,
  phon:(face:brow:Brow,
        mouth:pict:'tu:',
        man:(ndh:hns_string,
             hsh:(hd:hamfinger23spread,
                  tl:[hamthumbacrossmod]),
             ori:(plm:(hd:hampalmu,
                       tl:e_list),
                  efd:(hd:hamextfingeru,
                       tl:e_list)),
             const:[hamshoulders,hamlrat])),
  syn:(premod:[(@ np(W,Gloss,Index,Sem,dual))],
        postmod:[],
        head:(quantif_lxm,agr:(num:(number:dual))),
        arg_st:[(sem:(index:Index,Sem))]),
  sem:(index:(Index),
        restr:[(sit:Ind3,
                reln:two,
                sense:Sense,
                args:[(index:Index,Sem)])]).
```

Although the noun as its agreement has a number (*num*) feature, and logically it has to agree with the *num* of the *quantif_lxm* in SYN, the repetition of the noun sign does not take place (see also section above).

Number incorporation in BSL has not been implemented.

6.1.4 Plurals in ALE

In BSL nominal plurals can be expressed in several different ways. Some plurals are formed by repeating the sign (usually three times), each repetition beginning at the location where the previous finished. Neither singular signs which involve internal repetition nor body anchored signs (ones which involve contact between the dominant hand and another part of the body) can be pluralized in this way. However, such signs can take a proform (a 'pronominal' handshape classifier) which can be repeated in a comparable fashion. Quantifiers, which occur before the noun in BSL, are also used for pluralization, but quantifiers can also be expressed as part of the internal construction of a sign. The distributive movement of the verb expresses that members of a group are involved individually in an action, but sweeping movement indicates the collective involvement of the whole group. Repetition of some verbs can mean either that one individual repeats the action or that many individuals do the same action (Sutton-Spence 99). Currently, of these possibilities of noun pluralization, we handle nouns, which can be repeated and/or used in combination with a sweep motion and non-repeatable ones with external quantifiers. However, for pluralization of the remaining group of nouns the feature structure design contains the relevant classifier information about the possible proforms (substitutors) for future development.

The current implementation of the *plural_principle* for nouns takes the SEM:COUNT information from the SEM component (COUNT in example (3),). The lexical item determines whether it allows plural repetition (or sweep movement in which case the feature is ALLOW_PL_SWEEP), if so, then the PHON:MAN:MOV:REPEAT feature (MOV = movement) is instantiated to the HamNoSys symbol expressing repetition in different locations and COUNT from the semantic input is propagated as the current value to the noun's SYN:AGR:NUM feature (in case of collective meaning the values are COLLORDIST:coll and MAN:MOV:REPEAT: [?]) while ALLOW_PL_REPEAT is instantiated to 'no'. In all other cases PHON:MAN:MOV:REPEAT remains uninstantiated. It has to be noted that the number of repetition is relevant up to 3, which corresponds to the actual number but three repetitions can also mean any number. Therefore NUMBER: dual has been introduced, in which case the repetition happens only twice.

```
plural_principle_noun(syn:(allow_pl_repeat:yes_loc_indiv_finite,
    head:(noun,agr:(num:Sg,per:Per))),
sem:count:(number:pl,collordist:distrib),
    syn:(allow_pl_repeat:no,
    head:(noun,
        agr:(num:(number:pl,
            collordist:distrib),
            per:Per))),
man:mov:[(repeat:[hamseqbegin,
hamrepeatfromstartseveral,
hammove,
hamseqend])])
if true.
```

Verb pluralization is handled in a similar way, however the repeated verb motion is only permitted if the index of the semantic role and the index of the appropriate complement are identical, i.e. the agreement is correct. Verbs also have to be informed, whether the source (SRC) or goal (GOL) location has to be repeated or swept, therefore a simple use of REPEAT like for the nouns, is not possible:

```
man:mov:[(repeat:R,
    src:(por:Index2,
        rep:R2,
        gl:Gloss,
```

```

        height:Heightobj,
        dist:Distobj),
    gol:(por:Index,
gl:Glosssubj,
rep:[hamseqbegin,hamrepeatfromstartseveral,
    hammover,hamseqend],
    height:Heightsubj,
    dist:Distsubj)]

```

After quantifiers the pluralization as described above does not occur. Therefore a *quantif_principle* has been introduced, which determines in the modifier rules that the noun's REPEAT feature's value is an empty HamNoSys list.

```

quantif_principle(quantif_lxm,PhonetN,man:mov:[(repeat:[ ])])
    if true,!.

```

6.1.5 Temporal Adverbs

6.1.5.1 Temporal adverbs in DGS

Temporal adverbs are of type *tempadv-lxm*:

```

tempadv-lxm := non-compound-lxm & modifier-sem &
[ SYN.HEAD tempadv,
  SEM.RESTR <! temporal-predication !> ].

```

Examples of temporal adverbs are NOW and TODAY:

```

today_1 := tempadv-lxm &
[ GLOSS <! "TODAY" !>,
  PHON [MOUTH [ PICT-DE "heute",
                PICT "hOYt@" ],
        MAN one-handed &
        [ HSH <! hamfinger2,hamthumbacrossmod,hamfingerstraightmod !>,
          ORI [EFD <! hamextfingero !>,
                PLM <! hampalmd !> ],
          MOV [MLOC <! hamchest !>,
                M-HNS <! hammoved,hamsmallmod,hamrepeatfromstart !>]]],
  SEM.RESTR <! [ TEMP temp_today ] !> ]].

```

There are also three subtypes of temporal adverbs *tempadv-incorp-lxm*, *tempadv-someseq-lxm*, *tempadv-seq-lxm*. The first group is described above in the section on incorporation. The *tempadv-someseq-lxm* has as PRECOMPS.SYN.HEAD *aboveten*, i.e. they appear in sequential order with the corresponding numbers only with numbers 11 and above. (These signs incorporate the numbers 1–10.)

```

tempadv-someseq-lxm := tempadv-lxm &
[ SYN [HEAD [ AGR.NUM #num ],
        PRECOMPS < [ SYN.HEAD aboveten & [ AGR.NUM #num ] ] >,
        POSTCOMPS < >,
        QPRO false ]].

```

The other temporal adverbs (e.g. SECOND, DAY) cannot incorporate number, they are called

```

tempadv-seq-lxm := tempadv-lxm &
[ SYN [HEAD [ AGR.NUM #num ],
        PRECOMPS < [SYN.HEAD det & [ AGR.NUM #num ] ] >,
        POSTCOMPS < >,
        QPRO false ]].

```

6.1.5.2 Temporal adverbs in BSL

Temporal adverbs belong to the type *adverbs*. Currently adverbs do not have any subtypes as only simple temporal adverbs (like NOW, TODAY, NEXT) are handled in the lexicon in a similar way to DGS. Adverbs are treated as sentence modifiers:

```
syn: (precomps: [],  
      postcomps: [( $\emptyset$  vp(sentence, Index1, Sem, Sg))]).
```

Simple adverbs tend to be used at the front of a sentence, therefore complements are determined as postcomplements in the lexical item.

6.2 Grammar

6.2.1 Question pronouns and gap threading

6.2.1.1 Literature review

Most of the literature on question pronouns is on question pronouns in ASL. In the literature, there are two different theories regarding the syntactical position of question pronouns in ASL. While some authors (e.g. Lillo-Martin 2000) argue that there is a leftward movement of the question pronouns to the beginning of the sentence, others (e.g. Neidle et al. 2000) believe that there is a rightward movement of the question pronouns to sentence-final position. The reduplication of question pronouns in ASL (i.e. the appearance of the question pronoun both at the beginning and at the end of a sentence) that occurs frequently can be explained with either of these theories.

6.2.1.2 DGS

6.2.1.2.1 Data collection

There is no comparable literature on question pronouns in DGS. We therefore collected our data in interviews with nine native signers who work at the IDGS. The following methods were used:

- We asked the informants to translate written German sentences into DGS. Note that we only asked those informants who felt comfortable with the written language.
- We described a situation in DGS and asked the informants to produce an adequate question in DGS.
- The informants were given a specific question pronoun and they were asked to produce as many different questions as possible using this question pronoun.
- We described a situation in DGS and then asked the informants a question in DGS. We wanted them to
 - a. answer the question; and to
 - b. judge its syntactical correctness, rating it: very good, acceptable / comprehensible, or wrong.

If a question was not answered correctly the judgment was not taken into consideration.

All methods were also used in a combined way and informants had the opportunity to correct their answers or to give additional information. All answers were transcribed in glosses, and some more complex interviews were recorded on video.

6.2.1.2.2 Analysis

In DGS, the verb occurs in the last position in a sentence, the arguments and modifiers occur before the verb. However, the analysis of the interviews showed that in a sentence with a question pronoun, this no longer holds true.

We differentiate between different types of question pronouns:

6.2.1.2.2.1 Question pronouns that modify the verb

The question pronouns that modify the verb (“WHEN”, “WHY”, “HOW”, “HOW-LONG” etc.) appear in final position of a sentence. We treated them as post-modifiers of the verb. In order to prevent other modifiers from appearing after the question pronoun, we introduced a new feature, QPRO of type *boolean*, which prevents all modifiers from applying to a verb already modified by a question pronoun.

- “who”, “whom”, “to whom” (“WER” in DGS): their grammatical category is a subtype of the person-reference-noun;
- “what” (“WAS” in DGS): its grammatical category is a subtype of the object-reference-noun.

For this group of question pronouns, the word order is

(O) V S(expressed by a question pronoun; e.g. CLEAN WHO? “who cleans?”); or

S (O₁) V O₂(expressed by a question pronoun; e.g. I GIVE BOOK WHO? “To whom do I give the book?”).

(Note that for the gap threading principle we did not consider another possible word order, namely WHO CLEAN WHO? which is the same as CLEAN WHO?)

In order to analyse the behaviour of the question pronouns as arguments of the verb, we divided the verbs into four classes, according to their agreement characteristics:

- fixed verbs,
- fully directional,
- partially directional, and
- orientational verbs (directional verbs, which, additionally, change the finger orientation when the agent is a non-first person).

For each category we chose test sentences in German and asked native signers to sign them. The responses were filmed, transcribed, and analysed. We extracted the descriptive rules for each pronoun in each argument position. As the behaviour of the question pronouns does not depend on the category of the verb it is used with, we obtained a single set of rules for all verbs. Once these rules were established, we tested them: We generated DGS sentences according to these rules, filmed the DGS sentences, and asked the native signers to judge their correctness. The majority of the signers were satisfied with the result. Finally, with this feedback we proceeded to implement the question pronouns in our HPSG grammar.

6.2.1.2.2.2 Other question pronouns

The DGS question pronoun “WOHER” (meaning in English “from where”) is an exception because it functions as a verb itself, meaning either “come from where” or “get from where”. Thus we introduced two different lexical entries for WOHER, both of them verbs, the first corresponding to the meaning “come from where” with one argument (the agent), and the second corresponding to the meaning “get from where” with two arguments (agent and theme).

6.2.1.2.2.3 Gap threading

In order to make the above sentence structure possible we implemented the mechanism of gap threading (introduced by Pereira 1981). This mechanism allows two elements (for example a verb and one of its arguments) to appear far from one another in a sentence, despite the existence of a syntactic dependency between them. In our grammar this is the case for a verb that takes a question pronoun as argument. To be able to build clauses with missing elements we introduced a new feature, called GAP, which keeps track of the missing elements. Although our data show that a DGS sentence may contain only one question pronoun, the value of the feature GAP is a list of *synsem-struct*. This way the value of the feature GAP may be the empty list if no argument is missing (in our case that means that no argument of the verb is a question pronoun).

All arguments that a certain lexical entry requires are specified in the PRECOMPS and POSTCOMPS lists of that entry. We want to allow some arguments to be neither in the PRECOMPS list nor in the POSTCOMPS list, but in the GAP list. There are many possibilities for a certain lexeme to move the arguments from the PRECOMPS or the POSTCOMPS list into the GAP list resulting in a rather large set of possible instances for this lexeme. It is not desirable to add all these instances to the lexicon. Instead, we implemented a set of lexical rules which, for an existing lexical entry, generate all possible assignments of arguments in the PRECOMPS, POSTCOMP and GAP list. We adapted a grammar rule which already exists in the HPSG model we followed, the Head-Filler-Rule. This rule provides a filler to a phrase – the head daughter of the rule – which has a non-empty GAP list. The Head-Filler-Rule applies only if the filler and the element specified in the GAP list of the head daughter unify. Finally, we needed one more specification in our grammar to ensure that a phrase gets all its arguments, namely that the feature structure of the “root”, which describes a correct DGS sentence, has an empty GAP list (together with empty PRECOMPS and POSTCOMPS lists).

These rules implement, in fact, the **Argument Realisation Principle** (ARP), which states that: ARG-ST = PRECOMPS + POSTCOMPS + GAP (ARG-ST is the list of all arguments of a lexical entry). Following is an example of such a rule (in this case the PRECOMPS list contains one element and GAP is the empty list):

```
arg-real-principle-PRE1:= word+arg_real &
[ GLOSS #gloss,
  SYN [      HEAD #head,
    QPRO #qpro,
    REFPT-IN #refptin,
    REFPT-OUT #refptout,
    GAP #gap,
    PRECOMPS < #precomps >,
    POSTCOMPS #postcomps,
    ARG-ST < #precomps . #postcomps > ],
  SEM #sem,
  PHON #phon,
  ARGS < word & [GLOSS #gloss,
    SYN [ HEAD #head,
      QPRO #qpro,
      REFPT-IN #refptin,
      REFPT-OUT #refptout,
      GAP #gap & <! !>,
      PRECOMPS < #precomps >,
      POSTCOMPS #postcomps,
      ARG-ST < > ],
    SEM #sem,
    PHON #phon ] > ].
```

We adapted a grammar rule that exists in the HPSG model and followed (Sag and Wasow 1999), the Head-Filler-Rule. This rule provides a filler to a phrase – the head daughter of the rule – which has a non-empty GAP list. The Head-Filler-Rule applies only if the filler and the element specified in the GAP list of the head daughter unify.

We first tried to implement the Argument-Realisation-Principle (ARP) with difference lists:

```
PRECOMPS    list-of-comps & [LIST #1, LAST #2],
POSTCOMPS   list-of-comps & [LIST #2, LAST #3],
GAP         list-of-comps & [LIST #3, LAST #4],
ARG-ST      list-of-comps & [LIST #1, LAST #4],
```

This approach is not appropriate for the case when an element in the middle of the PRECOMPS or POSTCOMPS list must be in the GAP list. We therefore changed the type of PRECOMPS and POSTCOMPS list. They are now lists of *synsem-struct* instead of difference-lists. At the lexeme level there is no relation between PRECOMPS, POSTCOMPS, GAP and ARG-ST. The ARP transforms a *word* into a *word+arg_real-pr*. ARP is implemented by means of lexical rules (see above). In the *gtypes*-file, all Head-Complement-Rules have a *word+arg_real* as H-daughter.

We had to prevent the question pronouns (as argument of the verb) from appearing in the “normal” position of the argument. For example the DGS sentence I WHAT OPENVERTICAL should not be accepted as a correct sentence; it should instead be I OPENVERTICAL WHAT. To this end we introduced a new feature: QPRO is a SYN-feature of type *boolean*, and with default value *false*. All question pronouns are QPRO *true*.

All head-comp-rules require a NH-daughter that is QPRO *false*. The Head-Filler-Rule requires a filler which is SYN.QPRO *true* (i.e., the GAP list serves only to encode question pronouns). The H-daughter of this rule is QPRO *false*. After the rule is applied, the resulting phrase must be SYN.QPRO *true* in order to avoid the rule from applying a second time because a DGS sentence may not contain two question pronouns.

We also modified the Head-Postmodifier-Rule. The H-daughter is QPRO *false*, and the resulting phrase (the mother of the rule) gets its QPRO value from the modifier. Therefore, if the modifier is a question pronoun (QPRO *true*) the phrase becomes QPRO *true*, and no more question pronouns (whether as arguments or as modifiers) are accepted in the sentence. Moreover, as the rule only applies if the H-daughter is QPRO *false*, no other modifiers may appear after a question pronoun in the sentence.

The new QPRO-Principle passes the value of the feature QPRO from the H-daughter to the mother of a rule. The rules Head-Filler-Rule and Head-Postmodifier-Rule are not object of this principle.

Finally, we needed one more specification in our grammar to ensure that a phrase gets all its arguments, namely that the feature structure of the “root”, which describes a correct DGS sentence, has an empty GAP list (along with empty PRECOMPS and POSTCOMPS lists).

6.2.1.3 BSL

Currently four interrogatives are handled in BSL: ‘where’, ‘how many’, ‘who’ and ‘what’.

```
syn:(precomps:[(@ vp(sentence, Index, Sem, Sg))],
      head:(interrogative,
            context:(context_in : list_context,
                    add_list   : [],
                    context_out: list_context,
```

```

        delete_list:[ ]),
        form:proform),
    allow_pl_repeat:no,
    allow_pl_sweep:no,
    postcomps:[ ],
    arg_st:[ (sem:(index:Index,Sem)) ]),

```

The above SYN structure reflects the fact, that all the question pronouns are treated as complements to a sentence. This is non-problematic for 'how many' and 'where', which are verb modifying question pronouns. Even 'who' the subject reference question pronoun can be handled in this way (with the difference that it does not have a complement), since the *proform* value guarantees the required sign order. The last complement is checked for the form (*checkpronoun principle* see below) and therefore the *last_complement_pron* rule is used to ensure that the last complement is always the last sign if it is a proform. The pronoun 'what' is more problematic as it cannot guaranteed that it is a last complement in the structure. Gap threading will be needed but this has not been implemented yet in BSL.

```

checkpronoun((noun,agr:Arg,form:proform)) if true,!.
checkpronoun((interrogative,agr:Arg,form:proform)) if true,!.

```

The associated non-manual (eyebrows) with questions comes from the input SEM structure's *mode* feature. If the input's *mode* feature value is *whqu*, than in *phon:face:brow:Brow* the Brow is instantiated to *furrowed*. This value is then passed through the generation sequence.

```

mode(decl,non_raised) if true.
mode(whqu,furrowed) if true.
mode(yesnoqu, raised) if true.

```

6.2.2 Extended verb model

6.2.2.1 EGO Verbs

There are different types of directional verbs; some have a fix beginning location, some a fix end location. Most directional verbs in DGS are fully directional, i.e. both beginning and end location depend on the location of agent and patient/source and goal in signing space.

There is, however, a group of fully directional verbs that are not “truly fully” but only “ego fully”, i.e. path movements between non-first and non-first person (non1–non1) are performed like movements between first and non-first person (1–non1).

We have therefore introduced a HEAD feature, EGO, of type *boolean* which is *true* for the “ego fully” verbs. The default value of this feature is *false*.

1-non1, non1-1

For these forms, the ego verbs behave like “truly fully” directional verbs, the only difference is that the first argument (agent) cannot be dropped.

Lexical rules: ego-non1-1, ego-1-non1

```

ego-non1-1 := lrule-lexeme &
[ SYN [ PRECOMPS < #1 & [ SYN.HEAD.ALLOW-DROP false ], #2 >,
  POSTCOMPS #post,
  HEAD [ FORM #form,
    AGR #agr,
    AUX #aux,
    EGO false ]],
  SEM #sem,
  GLOSS #gloss,

```

```

PHON #phon,
ARGS < actpat-dir-verb-lxm &
  [ SYN [ PRECOMPS <#1 & phrase &
                                [SYN.HEAD pr-noun & [AGR.PER non-first]],
                                #2 & phrase &
                                [SYN.HEAD pr-noun & [AGR.PER first]] > ,
        POSTCOMPS #post,
        HEAD [ FORM #form,
              AGR #agr,
              AUX #aux,
              EGO true ]],
        SEM #sem,
        GLOSS #gloss,
        PHON #phon ] >].

```

non1-non1

For the non1-non1 form, the ego verbs behave like end-directional verbs. As the ego verbs are defined as fully directional, we have to modify the PHON feature of the non1-non1 form of an ego verb, i.e. the source of the movement must be the location of the first person.

Lexical rule: egoactpat-to-enddir

```

egoactpat-to-enddir := lrule-lexeme &
[ SYN [PRECOMPS < #1 & [ SYN.HEAD.ALLOW-DROP false ], #2 >,
      POSTCOMPS #post,
      HEAD [ FORM #form,
            AGR #agr,
            AUX #aux,
            EGO false ]],
      SEM #sem,
      GLOSS #gloss,
      PHON [MOUTH #mouth,
            SHLD #shld,
            BODY #body,
            HDMOV #hdmov,
            EYEG #eyeg,
            FACE #face,
            SUBEVENTS #subevents,
            MAN [ NDH #ndh,
                  HSH #hsh,
                  ORI #ori,
                  CONST #const,
                  WEAK-HAND #weak-hand,
                  REFPT #refpt,
                  MOV [ SRC.FLOC <! hamchest,hamclose !>,
                        GOL #gol,
                        REPEAT #repeat,
                        HSHF #hshf,
                        ORIF #orif,
                        PTH-MOD #path-mod,
                        FOB #fob,
                        PATH-MOV #path-mov ]]],
      ARGS < actpat-dir-verb-lxm &
        [SYN [PRECOMPS < #1 & phrase &
                                [SYN.HEAD pr-noun & [ AGR.PER non-first]],
                                #2 & phrase &
                                [SYN.HEAD pr-noun& [AGR.PER non-first]] > ],
        POSTCOMPS #post,
        HEAD [FORM #form,
              AGR #agr,

```

```

        AUX #aux,
        EGO true ]],
SEM #sem,
GLOSS #gloss,
PHON [MOUTH #mouth,
      SHLD #shld,
      BODY #body,
      HDMOV #hdmov,
      EYEG #eyeg,
      FACE #face,
      SUBEVENTS #subevents,
MAN [NDH #ndh,
     HSH #hsh,
     ORI #ori,
     CONST #const,
     WEAK-HAND #weak-hand,
     REFPT #refpt,
     MOV [GOL #gol,
          REPEAT #repeat,
          HSHF #hshf,
          ORIF #orif,
          PTH-MOD #path-mod,
          FOB #fob,
          PATH-MOV #path-mov]]] > ].

```

All “ego rules” change the value of the feature EGO – which is initially *true* – into *false*. In order to prevent an ego verb to move between non-first and non-first person (non1–non1), we add a constraint to the root, namely that a correct sentence must be [HEAD [EGO *false*]].

6.2.2.2 Aspectual inflection

6.2.2.2.1 Literature review

There is some literature on a number of sign languages on “aspectual verb modulations”. Klima/Bellugi (1979) give the first comprehensive account on aspectual modulation in ASL. They claim that there are eleven aspectual modulations on adjectival predicates which express onset, duration, frequency, recurrence, permanence, or intensity of states or events (pp. 247-271). In addition, they document several morphological processes in order to express the “grammatical category of temporal aspect and focus” (p. 291). Taking all these different possibilities into account, there are 16 possible modulations for verbs in ASL which they call: Predispositional, Susceptative/Frequentative, Continuative, Incessant, Intensive, Iterative, Approximative, Resultative, Protractive, Habitual, Inceptive, Durational, Continuative, Facilitative, Augmentative. These are expressed in ASL through changes in the way the hands move or in the number of repetitions, among many other things.

Bergman (1983) for Swedish SL and later Brennan (1992) for BSL show that different types of repetition (fast vs. slow), performed on either durative or punctual verbs, result in different verb modulations:

	slow redupl.	fast redupl.
durative vb	Continuative	Durational
punctual vb	Iterative	Habitual

In addition to these, there are some other possible modulations, including Inhibitional/Inceptive (Liddell 1984: Unrealized-inceptive).

Emmorey (1991) also mentions different “dynamic contours” for Continuative, Habitual, Intensive, and Iterative in ASL; the same modulations are called “aspect inflections” in Valli/Lucas (2001).

The problem with all of these sources that they all quote the same examples. Even if the data is not from ASL, the same verbs are used to illustrate certain modifications. This raises the question whether or not these modifications work for all other verbs in the same group. None of the authors state any exceptions, and there is no reference to possible phonological or semantic/syntactic restraints.

6.2.2.2.1.1 DGS

To our knowledge, there is no literature on aspect marking in DGS.

In the filmed DGS narrations of the BJ story, there are only very few occurrences of aspectual changes, all expressing Continuative/Durative. The three examples in the BJ data are:

WALK WALK WALK

STIR STIR STIR

STIR TALK STIR TALK.

One other interesting example is

SEARCH WHERE WHERE WHERE WHERE

where the aspect is marked on the question pronoun and not on the verb itself.

Another occurrence of an expression of Continuative/Durative is

SIGN TALK CHAT SIGN TALK

where the duration is expressed by using different signs with the same/similar meaning.

In informal interviews with native signers, they said that the subtle modifications documented by Klima/Bellugi (1979) in particular, cannot be found in DGS but are rather expressed with separate/additional signs. However, Habitual, Continuative, Iterative, and Durative do occur. They all felt that “aspectual” inflections are not always systematic. Inflection may work with one verb but not with another, and it may work in one context but not in another. Very often, temporal information needs to be added in separate signs (adverbs).

More research is needed to find out how regularly these modifications do occur (as modifications of verbs and not expressed with extra signs), whether there is a systematic correspondence between the phonological form of a verb, or its semantics, or its syntactic behaviour and the possibility to make inflectional modifications.

6.2.2.2.1.2 BSL

Information on BSL aspectual marking was ascertained by interview with a deaf colleague at Bristol. The following summarises some of the more significant generalisations which may inform the extension of the BSL sign generation system.

Many of the aspectual inflections are for verbs that describe what is happening. [By this presentable actions are meant, where the sign mimics the action sufficiently closely for the "inflection" to be isomorphic]. It is less apparent that such inflectional marking appear within informative descriptions typical of sign language translations of TV news reports. Therefore the areas of occurrence for different aspect are different.

Habitual

i Classic example of GO and GO-HABITUALLY.

E.g.: DEAF-CLUB GO-HABITUALLY/REGULARLY

The movement involves small, fast circles and combines semantic notions durational with something happening often.

ii Instead of saying REGULAR DRINKING (i.e. of alcohol) [where DRINK is repeated twice] one could sign DRINKING-REGULAR [repeated four times with a regular head nod].

iii If one drives regularly in a car, DRIVE would have comparable small circular movements. WORK D RIVE [with circular movement] means regularly use the car.

OFTEN and AGAIN can be used moving through space for expressing habituality.

Perfective

BEEN:

Most aspectual markers come after the verb but BEEN can appear prior to the verb. One could sign THINK BEEN but the meaning would be different - it suggests that the thinking went on longer before finishing so it is more emphatic, that one had thought something through for a while and then it finally finished. BEEN THINK is more a simple time marker that the thinking is finished.

In a question like EAT BEEN however (meaning "have you eaten") it is more of a neutral question. BEEN EAT? is more like if one is not sure and asks more emphatically. It is more linked to uncertainty where facial expression is of course important.

Continuative/durative

CARRY-ON could be used to express duration. But for example in CARRY-ON DRIVING something like FAR-FAR would replace CARRY-ON. There is also a southern sign with similar meaning, with two clenched fists instead of the two 'c' hands. In Scotland the equivalent would be two 'c' hands at shoulder height moving forwards. Used like WORK CARRY-ON, but one could not use this sign for DRIVE CARRY-ON, only for WORK or LIVE or similar concepts. For DRIVE one would use FAR-FAR. If somebody was watching the petrol. Gauge going down and was desperately carrying on, one might use CARRY-ON like in CAN CARRY-ON because that would focus on and emphasise that exact idea. But one could also have the car proform creeping slowly along with a painful facial expression - which would have the same meaning. It is showing emotion again.

This continuous marker would not be for states like HAPPY or ILL. The fist across the palm like in REGULAR would be better or one could use the time period over from shoulder forward like

HAPPY PERIOD-UP-TO-THIS-POINT. BEYOND-THIS-POINT-FORWARD CHANGE WHAT? (i.e. I was happy for a while, then something happened and all that changed)

READ-FOR-A-LONG-TIME -- READ is moved forward along the shoulder time-line A.

IGNORE-WHILE-READING -- nd hand holds READ, while dom hand signs IGNORE.

These verbal modifiers have not been incorporated within the existing sign language grammar.

6.2.3 Signing space planning

6.2.3.1 Sign language phenomena

In sign languages when new referents are introduced into a discourse, they can be assigned a location in signing space (the area in front of the signer, which starts at waist level close to the body and ends above the head vertically and at arm's length in any direction horizontally).

This syntactic signing space is also used for other grammatical structures, which express movement between two discourse referents (directional verbs):

- i For example, 'I put a mug in the sink'. In this case the signer has a mental image of the positions of the 'mug' and the 'sink'. After the signer has placed the objects accordingly in her signing space the directional verb PUT is signed between those two points, 'sink' being the end position, where the 'mug' is placed after the movement.
- ii Some other directional verbs however do not have their end point at the located object, but approximately halfway between the start position and the position of the object. For example, the sign SEE starts at eye level of the signer and the movement is only directed towards the object that is seen and in which case the object is not replaced.
- iii There is group of directional verbs, which have full inflection, but whose start position is always at a higher horizontal level than the end position. For example, in the sign ASK the movement starts at chin level but finishes at breast level and is only directed toward the placed object.

Another grammatical structure that is related to the syntactic space is the use of pointing, which is known as indexing. Its function can be similar to either the definite determiners and the pronouns in English. This means pointing with the index finger toward the placed object. In this case it will be the direction of the finger that is related to position of the referent.

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 is the task of the signing space planner component in the generation system to model appropriate signing space planning strategies.

6.2.3.2 Signing space definitions

The Hamburg Notation System (HamNoSys) is a well-established phonetic transcription system for sign language, which comprises of more than 200 iconically motivated symbols. (Prillwitz et al. 1989). ViSiCAST is the first project to use HamNoSys for storing the phonetic form of individual signs in the lexicon and for combining signs into sign language utterances and for using it to drive avatars from this notation

HamNoSys 4.0, the enhanced version of the original system (further refinements of manual aspects and encoding of non-manual aspects) is the basis for the design of the signing space planner. Signing space locations (allocation map) can be defined by the combination of basic HamNoSys symbols, which can express a unique point in the signing space. On a horizontal line across the signer's body there are five such positions are defined: beside the body part to the left, at the body part to the left, at body part, beside the body part to the right and at the body part to the right. On the vertical plane the height of the position is given by the body parts: shoulder level, chest level and stomach level. The depth of the space is defined by the extension of the arms: close to body, neutral space and extended arms

The nouns and the verbs of group (i) above can be characterised in terms of this allocation map. The second and third groups in (ii) and (iii) however need an extension of the map, as each position has to be associated with a HamNoSys direction that indicate movement towards a specific signing space location. These types of movements are the combinations of the following movement primitives (HamNoSys symbols): left, right, away from body, towards body, upwards, forwards, downwards, horizontal and slanted.

The third extension to the allocation map is needed for indexing. Indexing does not require movement and therefore it can be defined neither by end positions nor by the direction of movement. The direction of the index finger should indicate the position that is referred to. In HamNoSys this can be done by the extended finger direction, which corresponds to the direction of the vector originating at the wrist, running along the length of the back of the hand and continuing in the direction pointed to by the fingers when fully extended (Prillwitz et al. 1989, version 2.0).

This initial map described in this section can be extended by further points at head level and intermediate points between two positions and finger directions to achieve a more refined structure of available points in space.

6.2.3.3 Incorporation within HPSG

6.2.3.3.1 Phonology

Sign variations such as the ones described above in (i) are encoded in the lexicon in a way, which reflects the possibility of full inflection, i.e. they are defined to refer to two locations. The model incorporates therefore two features SRC (source) and GOL (goal) in the HPSG PHON (phonology) part. Under MOV (movement) SRC and GOL themselves have further features such as HEIGHT and DIST (distance).

Verbs in group (ii) and (iii) do not have the object's position as the end point of the movement, therefore their end point is determined by another feature, which is a movement associated with the position in GOL. This movement is to find under the SYN feature called HAFWAY_MOV in the allocation map.

The final phonetical surface realisation of the sign is achieved by the concatenation of the HamNoSys symbols. The lexicon stores forms (left hand side of a lexical entry), which contain variables, whose values are instantiated after the allocation of the positions happened and therefore the values of HEIGHT and DIST are known. The following is an example for a lexical entry, which is a variable sign with full inflection. The first brackets contain the gloss, the second the non-manual features and the third the mouthing transcribed in SAMPA and all manual features. All values beginning with a capital letter are uninstantiated:

```
[[take],  
[Brow],  
['teIk', Nhd, Handshape, ExtendedFingerDirection, Palm, Heightobj, Distobj,  
Pluralistributive, hamreplace, ExtendedFingerDirection, Palm, Heightsubj,  
Distsubj, Pluralcollective]
```

6.2.3.3.2 Syntax

The argument structure and the agreement components of the SYN structure determine conditions under which signs can be combined into a grammatically correct physical realisation. Therefore all the features, which are required for the correct inflection of the signs based on location, are added to the SYN feature under the HEAD. In the SYN:HEAD a CONTEXT

feature is introduced, which consist of the CONTEXT_IN, CONTEXT_OUT, ADD_LIST and DELETE_LIST features.

The CONTEXT_IN contains all the positions, which are available in the allocation map and reflects the state at a given point of the generation with all occupied and free positions. The following is an example for the description of a position as described above using HamNoSys mnemonics.

```
syn:head:context:context_in:
  [(glossref:[(glossr:i,ref:Ind1)],
  halfway_mov:[hammovei],
  locat:(locatplm:[hampalmr],locatefd:[hamextfingeri]),
  distance:[hamclose],
  heights:[hamchest])
```

An ADD_LIST defines new positions required by a verb to specify its movement's start and end positions. Even in case of fixed signs it is the verb (or possibly an adjective or a preposition), which determines the position of a noun, that can be referred to. Therefore this feature is restricted to verbs, prepositions and adjectives. The DELETE_LIST is required for verbs of movement to express the relocation of an object. As previously shown above, the sign PUT relocates its object to a new location after which the old location is not valid any more. This list contains the information of the position that has to be freed. The CONTEXT_OUT list is the outgoing list after the new positions have been allocated or the not valid ones are deleted from the original CONTEXT_IN list.

In the process of allocation the ADD_LIST is compared with the CONTEXT_IN list. If the verb's argument already has a position then no new position is required.

Different objects with the same gloss are indexed differently. If the verb expresses relocation of a certain object, then a second position is allocated to it. This second position however has to be controlled by other arguments or even adjuncts, as the new location usually cannot be an arbitrary one. In case of PUT the object has to have the same position as the location where it is put.

The updated CONTEXT_IN list undergoes the process of deletion, when the elements of the DELETE_LIST are deleted from that list resulting in the final CONTEXT_OUT for that grammar rule.

6.2.3.3.3 Threading in the ALE implementation of HPSG

ALE's internal generation algorithm is semantic head driven (SHD) - a natural approach to generation with HPSG. It operates by discovering a pivot, which is the lowest node in a derivation tree that has the same semantics as the root.

Grammar rules are divided into two kinds, chain rules (which have a semantic head - whose head daughter's logical form is identical to the logical form of the mother) and non-chain rules (which have no semantic head or are lexical entries). The pivot is identified as the mother node of a non-chain rule operating in a top-down fashion. After the pivot has been found, it generates bottom-up using chain-rules to connect the semantic-heads to the pivot (Carpenter 1999).

It is important that the allocation of the positions happens at a stage when all objects (their glosses and index) are known, because of the relativeness of the positions of the objects to each other (e.g.: The final position of an object after replacement might depend on the subject ('I take the mug.'). If the subject is generated later then the object and the object's allocation has already happened, the subject's position might be allocated wrongly to unify the subject with the object

as required by the ADD_LIST. (In the example sentence 'I' would be unified with an arbitrary position of the 'mug' although the subject is already in the context list with a fixed position for the first person singular pronoun). If the positions are allocated at a later stage, when all complements of the verb are known, it is possible to control the occurrences in the CONTEXT_IN list and the ADD_LIST relative to each other. Thus, the allocation and deletion is restricted to grammar rules, in which the semantic head has no further complements.

It is vital that in the generation algorithm that the updated valid context is used at each point of the generation. Because the generation algorithm works in top-down fashion to find the pivot, the CONTEXT_IN list has to be passed down unchanged to the bottom. Therefore the updated CONTEXT_OUT cannot simply be passed up to the next level in the mother as the new CONTEXT_IN, because in some cases it would cause a contradiction between CONTEXT_IN of the mother and daughter. In chain-rules it means that the daughter needs the CONTEXT_IN information from its head-daughter, but after generating this daughter CONTEXT_IN may be instantiated to a modified CONTEXT_IN, which would cause it fail.

After generating the daughter and head as phrases the CONTEXT_OUT of each may contain an updated allocation map itself with new allocated positions. Therefore if only one of the new CONTEXT_OUT was passed up to the mother, some information would get lost.

Example 1:

```
Rule [phrase, SYN:HEAD:CONTEXT_IN:Z,
      SYN:HEAD:CONTEXT_OUT:Z ]
  -> H [phrase, SYN:HEAD:CONTEXT_IN:B,
        SYN:HEAD:CONTEXT_OUT:Z ],
      [phrase, SYN:HEAD:CONTEXT_IN:A,
        SYN:HEAD:CONTEXT_OUT:W ]
```

Therefore in chain-rules, which still have complements to generate but have no allocation to do, the mother's SYN:HEAD:CONTEXT_OUT value is W preserving the original SYN:HEAD:CONTEXT_IN. The CONTEXT_OUT of the mother in such rules will be the basis for the allocation on higher levels of the generation.

In chain rules, which do allocation, a more complicated picture emerges. In the grammar daughter nodes can be phrases or even sentences themselves (e.g.: verbal or modifier phrases). This means that they also need the context when generated. However, in such cases both the daughter and the semantic head (the semantic head can be a phrase itself) might have changed the original context list in a similar way as described above. Both new allocation maps have to be preserved for the next allocation in that rule. For such cases the union of the two CONTEXT_OUT lists is created. The union of Z and W results in Y, which is used as the ingoing context for the allocation. After allocation and deletion the result X is the mother's CONTEXT_OUT value as it is shown in Example 2.

Example2:

```
Rule [SYN:HEAD:CONTEXT_IN:A, SYN:HEAD:CONTEXT_OUT:X ]
-> H [ [ SYN:HEAD:CONTEXT_IN:A,
        SYN:HEAD:CONTEXT_OUT:Z ], [ SYN:HEAD:CONTEXT_IN:A
        SYN:HEAD:CONTEXT_OUT:W ] ]
```

6.2.4 Further grammatical phenomena implemented in ALE

6.2.4.1 Prodrop

Sign languages typically contain verbal signs which allow pronoun drop (prodrop), where one or more of the subject, object or indirect object (or actor, theme, addressee respectively) are omitted and incorporated within the sign for the verb itself. The actor and addressee are included within the sign for the verb as starting and/or end position of the movement (so-called directional verbs). In the case of a direct object pronoun the handshape of the sign for the verb is inherited from the object/theme (so called classifier proforms). This phenomenon reflects a similar relation between rich agreement and non-overt expression of subject/object pronomina (Bos 1993), as in many languages, such as Italian and Hungarian. Indeed prodrop is found also in Chinese, which allows for 'topic-drop' without such rich morphology.

The non-overt realisation of the pronomina (prodrop) is catered for by an empty lexical entry whose LHS is instantiated to an empty list and has non-instantiated RHS feature structure values. When the complement rules are processed, the prodrop principles check the semantic head for the values of subject and object prodrop features in all three persons. The possible values are *can*, *can't* and *must*. If it is *must*, the empty lexical item is chosen, that is not of type *word* but *dropped* to avoid ambiguity. The feature structure for such a lexical item is looked up in the lexicon using the RELN feature in the SEM component, (in fact to achieve this we drop out of ALE into in-line Prolog and use its ALE representation of lexical entries using the ALE operator --->¹). In this way, the required SYN information of the empty string, which has to be unified with the complement information of the verb, is instantiated. This is an important step, as the verb may need the index of the noun for start and end position of the movement or the classifier information for the handshape as discussed above. If the prodrop value is *can't*, generation proceeds normally, generating the daughter in the usual way for separate signs. If the value is *can*, both solutions are generated, however a preferred order is realised by arranging the order of Prolog predicates accordingly.

Anaphora, which are resolved in the DRS (he, she, it, they, I, you) are labelled as *topic* in the DRS structure. TOPIC feature in HPSG has two values *topic* and *comment*. The arguments, which have the value *topic* coming from the DRS are dropped. Another possibility to handle the topic is indexing but this is not a general solution as directional verbs often do not require additional indexing. Therefore currently we adopted the prodrop mechanism to topical pronouns by default.

6.2.4.2 Prepositional constructs as adjuncts expressing locality

A *last_postcomplement_adjunct* rule has been added to the grammar. This rule handles prepositional phrases, which express locality like *on the table* or *in the cupboard*. The signs which express this locality tend to be signed at the beginning of a signed unit. This rule

¹ ALE supports empty categories, however they could not be used to our purposes of prodrop. Empty categories in ALE are declared as lexical entries in a special format, therefore they suffer from a similar deficiency as lexical. Inheriting syntactic information from another lexical entry would not be possible in this way without duplicating lexical entries and therefore increasing the size of the static lexicon. We also considered Wilcock's suggestion (Wilcock 1998) to eliminate empty categories as non-preferable entities in the theory. His ProFIT/SAX/SGX implementation uses a Complement Extraction Lexical Rule as proposed by (Pollard and Sag 1994), which is not implementable in our grammar, because of the general problems with lexical rules. Another waywas to write a general lexical item with very few instantiated values, and include strict constraints in the grammar rules by goals. Currently, this solution has been suspended as a non-general and non-elegant solution to the prodrop problem.

determines that the local adjunct is the first in a given sequence, which is ensured by the fact that the semantic daughter's type is *sentence*. The semantic head has to be a *prep_lxm* (preposition) and its type is *phrase*.

The *last_postcomplement_adjunct* rule is currently inflexible as it accepts only one adjunct, but it can be used in combination with temporal adverbs (see adverbs in Section 6.1.5.2).

7 Achievements and limitations

Deliverable D5-4 is a prototype system demonstrating that unrestricted English text can be prepared for sign presentations. The resulting system demonstrates

- (i) that the sign lexicon and sign formation rules permit synthesis of readable signs for significantly sized lexicons of concepts.
- (ii) that the SL signing space formulator can appropriately handle co-reference as pointing and agreement with directional verbs within a model of three dimensional signing space.
- (iii) that pronominal and co-reference within an English text can be handled appropriately to facilitate the formulator.
- (iv) that aspects of the English tense system, temporal and locative complement phrases can be converted to the appropriate semantic form from which synthesis can begin.

A methodological problem arises when visualising the synthesised content, because it is not always obvious whether deficiencies in the generated signs are the consequence of inadequacies in the grammar or due to misinterpretation of the HamNoSys phonological description within the backend avatar software.

The achievement of Deliverable D5-4 is that it demonstrates the potential for preparation of sign language presentations from English text via a Discourse Representation Structure semantic representation. The subsystems which analyse English text utilise a combination of existent and internally developed natural language processing components. Moreover, it uniquely embodies the first ever formulations of sign language grammars in sufficient detail to support synthetic generation of signing and the integration of this technology with an executable interpretation of the phonological notation which drives a virtual human.

8 References

- Albertsen, Karen. 1985. Verbal morphology in Danish Sign Language. In *SLR' 83: Proceedings of the 3rd International Symposium on Sign Language Research*, Rome, June 22-26, 1983, ed. William C. Stokoe and Virginia Volterra, 210–216. Rome: CNR; Silver Spring: Linstok Press.
- Bergman, Brita. 1983. Verbs and adjectives: Morphological processes in Swedish Sign Language. In *Language in sign: An international perspective on sign language*, ed. Jim Kyle and Bencie Woll, 3–9. London: Croom Helm.
- Boyes Braem, Penny. 1995 (3rd ed). *Einführung in die Gebärdensprache und ihre Erforschung*. Hamburg: Signum.
- Brennan, Mary. 1981. Grammatical processes in British Sign Language. In *Perspectives on British Sign Language and deafness*, ed. Bencie Woll, Jim Kyle and Margaret Deuchar, 120–135. London: Croom Helm.
- Brennan, Mary. 1992. The visual world of British Sign Language: An introduction. In *Dictionary of British Sign Language/English*, ed. David Brien, 1–133. London: Faber and Faber.
- Carpenter, B., and G. Penn. 1999. *The Attribute Logic Engine. User's Guide. Version 3.2 Beta*. Bell Labs, 1999.
- Cormier, Kearsey. 1997. Locus agreement in American Sign Language: An HPSG analysis. Paper presented at the International Conference on HPSG Cornell University July 18-20, 1997.
- Emmorey, Karen. 1991. Repetition priming with aspect and agreement morphology in American Sign Language. *Journal of Psycholinguistic Research* 5 (20): 365–388.
- Fischer, Susan D. 1973. Two processes of reduplication in American Sign Language. *Foundations of Language* 9: 469–480.
- Hanke, Thomas. 2002. iLex - A tool for sign language lexicography and corpus analysis. In: *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas de Gran Canaria, Spain, ed. Manuel González Rodríguez and Carmen Paz Suarez Araujo. Vol. III, 923–926. Paris: ELRA.
- Hanke, Thomas, and Siegmund Prillwitz. 1995. syncWRITER: Integrating video into the transcription and analysis of sign language. In *Sign Language Research 1994: Proceedings of the Fourth European Congress on Sign Language Research*, Munich, September 1-3, 1994, ed. Heleen F. Bos and Gertrude M. Schermer, 303-312. Hamburg : Signum.
- Hanke, Thomas, Reiner Konrad, and Arvid Schwarz. 2001. GlossLexer – A multimedia lexical database for sign language dictionary compilation. *Sign Language and Linguistics* 4(1/2): 161–179.
- Kamp, H & Reyle, U, 1993, *From Discourse to Logic*, 2, Kluwer.
- Kennedy J., Boguraev E. 1996. Anaphora for Everyone: Pronominal Anaphora Resolution without a Parser, In: *Proceedings of the 16th International Conference on Computational Linguistics COLING'96*, Copenhagen, Denmark, 5-9 August 1996.
- Klima, Edward S., and Ursula Bellugi (eds.). 1979. *The signs of language*. Cambridge, Mass., London: Harvard University Press.

- Liddell, Scott K. 1984. Unrealized-inceptive aspect in American Sign Language: feature insertion in syllabic frames. In *Papers from the 20th Regional Meeting, Chicago Linguistic Society, Chicago 26-27 April 1984*, ed. Joseph Drogo, Veena Mishra, and David Testen, 257–270. Chicago: Chicago Linguistic Society.
- Lillo-Martin, Diane. 2000. Early and late in language acquisition: Aspects of the syntax and acquisition of Wh-questions in American Sign Language. In: *The signs of language revisited : an anthology to honor Ursula Bellugi and Edward Klima*, ed. Karen Emmorey and Harlan Lane, 401–413. Mahwah, NJ : Erlbaum.
- Neidle, Carol, et al. 2000. *The syntax of American Sign Language: Functional categories and hierarchical structure*. Cambridge, Mass.: MIT Pr.
- Newkirk, Don. 1998 [1981]. On the temporal segmentation of movement in American Sign Language. *Sign language and linguistics* 2 (1): 173–211.
- Pereira, Fernando. 1981. Extraposition grammars. *Computational Linguistics* 7(4): 243–256.
- Pollard, Carl, and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. Stanford: CSLI.
- Prillwitz, S., Leven, R., Zienert, H., Hanke, T., Henning, J., others. 1989. HamNoSys Version 2.0; Hamburg Notation System for Sign Languages - An Introductory Guide. *International Studies on Sign Language and the Communication of the Deaf, Volume 5*. Hamburg. Signum.
- Sag, Ivan A., and Thomas Wasow. 1999. *Syntactic theory: A formal introduction*. Stanford: CSLI.
- Supalla, Ted, and Elissa L. Newport. 1978. How many seats in a chair? The derivation of nouns and verbs in ASL. In *Understanding language through sign language research*, ed. Patricia Siple, 91–132. New York, San Francisco, London: Academic Press.
- Sutton-Spence, R., Woll, B. 1999. *The Linguistics of British Sign Language. An Introduction*. University Press, Cambridge.
- Valli, Clayton, and Ceil Lucas. 2000 (3rd ed). *Linguistics of American Sign Language: An introduction*. Washington, D.C.: Gallaudet University Press.
- Wilcock, G. 1998. *Natural Language Generation with HPSG*. PhD. Manchester.

9 Appendices

9.1 Appendix 1: DRS BNF / Ontology

DRSmultsent ::= [*VariableBindings* , [*DRSsent* , *DRSsent*]]

VariableBindings ::= [*NominalAttributiveProposition* / *NominalAttributiveProposition*]
(currently only provision for coreference of nominals, to consider later anaphoric relationships with events, etc)

DRSsent ::= *drslabel* : [*VariableList* , [*SententialProposition* , *LabeledPropositions*]]

SententialProposition ::=

ImpVar : imperative (*Evar*) |
QuVar : yesno ([*Evar* | *Svar*]) |
QuVar : *whPred1* ([*Evar* | *Svar*]) |
QuVar : *whPred2* (*Var*)
QuVar : comment (*Var*)
comment() for declaratives

whPred1 ::= { **when, how, why** }

whPred2 ::= { **who, what, which** }

VariableList ::=

To consider format
List of all variables / labels used in the DRS
Possibly segmented into different categories for ease of extraction.
Currently we only generate variables v_n and set_n in the variable lists.

LabeledPropositions ::=

LabeledProposition |
LabeledProposition , *LabeledPropositions* |
DRS □ *DRS*

(last production may have undesirable consequences, permitting
LocationalPropositions, *CollectivePropositions*, *ReferentRelationProposition*,
TemporalRelationProposition as antecedents - possibly need to subcategorise.)

LabeledProposition ::=

QuantifiedVariable |
EventProposition |
StateProposition |
TemporalProposition |
LocationalProposition |
NominalAttributiveProposition |
AttributiveProposition |
CollectiveProposition |

NumericalQuantifiedDefinition |
ReferentRelationProposition |
TemporalRelationProposition

EventProposition ::-
Evar : *Proposition*

StateProposition ::-
Svar : *Proposition*

TemporalProposition ::-
Tvar : **time** (*SvarOrEvar*) |
Tvar : *TemporalPredicate* (*SvarOrEvar*, *Var*)
(*Tvar* denotes an interval)

TemporalPredicate ::-
in | **on**

LocationalProposition ::-
Lvar : *LocationalPredicate* (*SvarOrEvar*, *Var*) |
Lvar : *DireactionalPredicate* (*SvarOrEvar*, *Var*)

DirectionalPredicate ::-
to | **from** | **into** | **out_of**

LocationalPredicate ::-
within | **on** | **above** | etc.

SymbolicLocationalProposition ::-
SLVar : *LocationalPredicate* (*SvarOrEvar*, *Var*)
/* following Hungarian nomenclature, these are for pseudo locational uses of prepositional phrases, 'in a hat', 'in a good humour', 'on the phone' – essentially a place holder for further thought. */

AttributiveProposition ::-
Avar : *Proposition*

NominalAttributiveProposition ::-
Nvar : *Proposition*

SetDefinition ::-
SetVar = *sumof*(*Var*)
SetVar = *sumof*(*Var*, *drslabel*)
/* Use of the box notation, potentially requires DRSs to be labeled for restricting predicates for plurals – see [1] p343.
requires further thought. */

CollectiveSet ::-
Cvar : *plural*(*SetVar*)

DistributiveSet ::-

Qvar : *Quant*(*Var*)
Cvar : plural(*Var*)

Quant ::-

every | some | few | many

QuantifiedSet ::-

size(*SetVar*) = *NumericValue*

NumericValue ::-

one | two | three | four |

TemporalRelationProposition ::-

Tvar *TemporalOperator* *Tvar*

TemporalOperator ::-

= | < | > | \sqsubset | \supseteq

(temporal precedence and inclusion)

(possibly others for completeness)

Proposition ::-

Predicate (*SetOrVarList*)

Predicate ::-

PredName . *Sense*

PredName ::-

A lexical name, what restrictions do we wish to impose if any here?

Sense ::-

An item identifying the sense of the lexical item and the specific sense according to that source (e.g. Wordnet.3)

SetorVarList ::-

SetOrVar |
SetOrVar , *SetOrVarList*

SetOrVar ::-

SetVar | *Var*

SvarOrEvar ::-

Svar | *Evar*

Where

Var \sqsubset {v1,v2,v3, }

Evar \sqsubset {e1,e2,e3, }

```

Svar   □ {s1,s2,s3, ..... }
Tvar   □ {t1,t2,t3, ..... }
Lvar   □ {l1,l2,l3, ..... }
SetVar □ {set1,set2,set3, ..... }
Qvar   □ {q1,q2,q3, ..... }
Nvar   □ {a1,a2,a3, ..... }
Avar   □ {attr1,attr2,attr3, ..... }
Cvar   □ {c1,c2,c3, ..... }
drslabel □ {drs1,drs2,drs3, ..... }
ImpVar □ {imp1,imp2,imp3, ..... }
QuestVar □ {qu1,qu2,qu3, ..... }

```

/* possibly rename Nvar and Avar variables as n1,n2,n3, etc and a1,a2,a3 respectively */

9.2 Appendix 2: Actual BJ Text Input

```

I saw an exciting video.
I tell the story.
I am a woman in the film.
I invited four friends.
I cook soup.
I put an orange in the blender.
The blender explodes.
The orange splashes onto my front.
I drink wine.
My mother phones me.
I am busy.
I have a party.

```

9.3 Appendix 3: BJ text DRSs generated

```

[ a ( 0 ) : I ( v ( 0 ) ) ,
  attr ( 0 ) : exciting ( v ( 1 ) ) ,
  a ( 1 ) : video ( v ( 1 ) ) ,
  t ( 0 ) : when ( e ( 0 ) ) ,
  t ( 0 ) < now ,
  e ( 0 ) : see ( v ( 0 ) , v ( 1 ) ) ]

[ e ( 1 ) : tell_story ( v ( 0 ) ) ,
  a ( 2 ) : I ( v ( 0 ) ) ,
  t ( 1 ) : when ( e ( 1 ) ) ,
  t ( 1 ) = now ]

[ s ( 0 ) : be_1 ( v ( 0 ) , v ( 5 ) ) ,
  a ( 4 ) : I ( v ( 0 ) ) ,
  a ( 5 ) : woman ( v ( 5 ) ) ,
  t ( 2 ) : when ( s ( 0 ) ) ,
  t ( 2 ) = now ,
  a ( 6 ) : film ( v ( 6 ) ) ,
  l ( 0 ) : in ( s ( 0 ) , v ( 6 ) ) ]

[ e ( 2 ) : invite ( v ( 0 ) , set ( 0 ) ) ,
  a ( 7 ) : I ( v ( 0 ) ) ,
  set ( 0 ) = sumof ( v ( 8 ) ) ,

```

```

size ( set ( 0 ) ) = four ,
t ( 3 ) : when ( e ( 2 ) ) ,
t ( 3 ) < now ,
a ( 8 ) : friend ( set ( 0 ) ) ,
c ( 0 ) : plural ( set ( 0 ) ) ]

[ e ( 3 ) : cook ( v ( 0 ) , greek ( 0 ) ) ,
a ( 9 ) : I ( v ( 0 ) ) ,
t ( 4 ) : when ( e ( 3 ) ) ,
t ( 4 ) = now ,
a ( 10 ) : soup ( greek ( 0 ) ) ]

[ e ( 4 ) : put ( v ( 0 ) , v ( 11 ) , v ( 12 ) ) ,
a ( 11 ) : I ( v ( 0 ) ) ,
a ( 12 ) : orange ( v ( 11 ) ) ,
t ( 5 ) : when ( e ( 4 ) ) ,
t ( 5 ) = now ,
a ( 13 ) : blender ( v ( 12 ) ) ]

[ e ( 5 ) : explode ( v ( 12 ) ) ,
a ( 14 ) : blender ( v ( 12 ) ) ,
t ( 6 ) : when ( e ( 5 ) ) ,
t ( 6 ) = now ]

[ e ( 6 ) : splash_onto ( v ( 11 ) , v ( 15 ) ) ,
a ( 15 ) : orange ( v ( 11 ) ) ,
t ( 7 ) : when ( e ( 6 ) ) ,
t ( 7 ) = now ,
attr ( 1 ) : my ( v ( 15 ) ) ,
a ( 16 ) : front ( v ( 15 ) ) ]

[ e ( 7 ) : drink ( v ( 0 ) , greek ( 1 ) ) ,
a ( 17 ) : I ( v ( 0 ) ) ,
t ( 8 ) : when ( e ( 7 ) ) ,
t ( 8 ) = now ,
a ( 18 ) : wine ( greek ( 1 ) ) ]

[ e ( 8 ) : phone ( v ( 17 ) , v ( 0 ) ) ,
a ( 20 ) : me ( v ( 0 ) ) ,
attr ( 2 ) : my ( v ( 17 ) ) ,
a ( 19 ) : mother ( v ( 17 ) ) ,
topic ( a ( 20 ) ) ,
t ( 9 ) : when ( e ( 8 ) ) ,
t ( 9 ) = now ]

[ s ( 1 ) : be_2 ( v ( 0 ) , v ( 20 ) ) ,
a ( 21 ) : I ( v ( 0 ) ) ,
attr ( 3 ) : busy ( v ( 20 ) ) ,
t ( 10 ) : when ( s ( 1 ) ) ,
t ( 10 ) = now ]

[ e ( 9 ) : have ( v ( 0 ) , v ( 22 ) ) ,
a ( 22 ) : I ( v ( 0 ) ) ,
a ( 23 ) : party ( v ( 22 ) ) ,
t ( 11 ) : when ( e ( 9 ) ) ,
t ( 11 ) = now ]

```

9.4 Appendix 4: BJ BSL HamNoSys generated

```
[ [ video ],
  [ non_raised ],
  [ hamsymmpar, hamfinger23spread, hamthumbacrossmod, hamextfingerdo,
    hampalmd, hamcircled ] ],
[ [ excited ],
  [ non_raised ],
  [ hamsymmlr, hamalternatingmotion, hamfinger2345, hamthumboutmod,
    hamfingerbendmod, hamextfingerl, hampalml, hamchest,
    hambetween, hamchest, hamlat, hamtouch,
    hamseqbegin, hammoveu, hammoved, hamseqend,
    hamrepeatfromstart ] ],
[ [ see ],
  [ non_raised ],
  [ hamfinger23, hamthumbacrossmod, hamextfingeru, hampalml,
    hameyes, hamlat, hamtouch, hammoveor ] ],
[ [ dropped ],
  [ ],
  [ ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ tell_story ],
  [ non_raised ],
  [ hamsymmlr, hamalternatingmotion, hamflathand, hamthumboutmod,
    hamextfingerul, hampalmul, hamparbegin, hammiddlefinger,
    hamfingertip, hamplus, hampalm, hamparend,
    hamclose, hamcirclel, hamparbegin, hammiddlefinger,
    hamfingertip, hamplus, hampalm, hamparend,
    hamclose, hamrepeatfromstart ] ],
[ [ dropped ],
  [ ],
  [ ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ in ],
  [ non_raised ],
  [ hamfinger2, hamthumbacrossmod, hamextfingero, hampalml,
    hamshoulders ] ],
[ [ film ],
  [ non_raised ],
  [ hamparbegin, hamflathand, hamextfingeru, hampalmd,
    hamplus, hamfinger2, hamthumbacrossmod, hamextfingerr,
    hampalmr, hamparend, hamparbegin, hamindexfinger,
    hamplus, hamwristback, hampalm, hamparend,
    hamswinging ] ],
[ [ be_1 ],
  [ non_raised ],
  [ ] ],
[ [ woman ],
  [ non_raised ],
  [ hamfinger2, hamthumbacrossmod, hamextfingeru, hampalmdl,
    hamlips, hamlat, hamseqbegin, hamtouch,
    hamindexfinger, hamfingertip, hamseqend, hamparbegin,
    hammoveo, hamsmallmod, hamseqbegin, hambrushing,
    hamlips, hamtouch, hamseqend, hamparend,
    hamrepeatfromstart ] ],
```

```

[ [ me ],
  [ non_raised ],
  [ hamfinger2, hamthumbacrossmod, hamextfingeril, hampalmr,
    hamchest, hamtouch ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ four ],
  [ non_raised ],
  [ hamfinger2345, hamthumbacrossmod, hamextfingeru, hampalmu,
    hamshoulders, hamlrat ] ],
[ [ friend ],
  [ non_raised ],
  [ hamsymmlr, hamparbegin, hamceeall, hamextfingerdl,
    hambetween, hamextfingerl, hampalml, hamplus,
    hamflathand, hamfingerbendmod, hamthumboutmod, hamextfingerr,
    hampalmr, hamparend, hamparbegin, hamhandback,
    hamplus, hampalm, hamparend, hamtouch,
    hammoved, hamsmallmod, hamrepeatfromstart ] ],
[ [ invite ],
  [ non_raised ],
  [ hamceel2, hamextfingeruo, hampalmd, hamlrat,
    hamshoulders, hamreplace, hampinchl2open, hamextfingeruo,
    hampalmd, hamchest ] ],
[ [ dropped ],
  [ ],
  [ ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ cook ],
  [ non_raised ],
  [ hamsymmlr, hamalternatingmotion, hamfinger2, hamfingerhookmod,
    hamextfingero, hampalml, hamseqbegin, hammovei,
    hamsmallmod, hammoveo, hamsmallmod, hamseqend,
    hamrepeatfromstart ] ],
[ [ soup ],
  [ non_raised ],
  [ hamfinger2, hamfingerhookmod, hamextfingerol, hampalml,
    hamshouldertop, hamcirclel, hamsmallmod, hamrepeatfromstartseveral ] ],
[ [ me ],
  [ non_raised ],
  [ hamfinger2, hamthumbacrossmod, hamextfingeril, hampalmr,
    hamchest, hamtouch ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ orange ],
  [ non_raised ],
  [ hamfinger2345, hamthumboutmod, hamfingerbendmod, hamextfingeru,
    hampalmdl, hamlips, hamlrat, hamclose,
    hamreplace, hamfist, hamrepeatfromstart ] ],
[ [ blender ],
  [ non_raised ],
  [ hamsymmpar, hamparbegin, hamfinger2, hamthumbacrossmod,
    hamextfingerd, hampalmd, hamplus, hamfinger2,
    hamthumbacrossmod, hamextfingeru, hampalmu, hamparend,
    hamparbegin, hamindexfinger, hamfingertip, hamplus,
    hamindexfinger, hamfingertip, hamparend, hamtouch,
    hamstomach, hamlrat, hamparbegin, hamcircleu,
    hamplus, hamcircled, hamparend ] ],

```

```

[ [ put ],
  [ non_raised ],
  [ hamceeall, hamextfingeror, hambetween, hamextfingerd,
    hampalml, hamstomach, hamlrbeside, hamreplace,
    hamceeall, hamextfingeror, hambetween, hamextfingerr,
    hamstomach, hamlrat, hamarmextended ] ],
[ [ dropped ],
  [ ],
  [ ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ blender ],
  [ non_raised ],
  [ hamsymmpar, hamparbegin, hamfinger2, hamthumbacrossmod,
    hamextfingerd, hampalmd, hamplus, hamfinger2,
    hamthumbacrossmod, hamextfingeru, hampalmu, hamparend,
    hamparbegin, hamindexfinger, hamfingertip, hamplus,
    hamindexfinger, hamfingertip, hamparend, hamtouch,
    hamstomach, hamlrat, hamparbegin, hamcircleu,
    hamplus, hamcircled, hamparend ] ],
[ [ explode ],
  [ non_raised ],
  [ hamsymmlr, hampinchall, hamfingerhookmod, hamextfingero,
    hampalmu, hamstomach, hamclose, hamparbegin,
    hammoveu, hamsmallmod, hamreplace, hamfinger2345,
    hamthumboutmod, hamextfingeru, hamparend, hammoveu ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ orange ],
  [ non_raised ],
  [ hamfinger2345, hamthumboutmod, hamfingerbendmod, hamextfingeru,
    hampalmdl, hamlips, hamlrat, hamclose,
    hamreplace, hamfist, hamrepeatfromstart ] ],
[ [ splash_onto ],
  [ non_raised ],
  [ hamsymmlr, hamfinger2345, hamthumboutmod, hamextfingeru,
    hampalmu, hamshoulders, hamlrat, hambetween,
    hamshoulders, hammoveui, hamshoulders, hamlrat,
    hambetween, hamshoulders, hamseqbegin, hamtouch,
    hamindexfinger, hamfingertip, hamseqend, hammoved,
    hamslow ] ],
[ [ dropped ],
  [ ],
  [ ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ wine ],
  [ non_raised ],
  [ hamfinger2, hamthumboutmod, hampinky, hamextfingeruo,
    hampalml, hamneck, hamparbegin, hamreplace,
    hamextfingeru, hammoveui, hamparend, hamlips,
    hamclose ] ],
[ [ drink ],
  [ non_raised ],
  [ hamceeall, hamextfingero, hampalml, hamchin,
    hamlrat, hamseqbegin, hamclose, hamthumb,
    hamseqend, hamreplace, hamextfingeru ] ],
[ [ me ],

```

```

[ non_raised ],
[ hamfinger2, hamthumbacrossmod, hamextfingeril, hampalmr,
  hamchest, hamtouch ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ mother ],
  [ non_raised ],
  [ hamparbegin, hamflathand, hamthumbacrossmod, hampinky,
    hamfingerhookmod, hamextfingerol, hampalmd, hamplus,
    hamflathand, hamextfingeror, hampalmu, hamparend,
    hamparbegin, hampalm, hamplus, hamindexfinger,
    hamfingertip, hamparend, hamclose, hammoved,
    hamsmallmod, hamparbegin, hampalm, hamplus,
    hamindexfinger, hamfingertip, hamparend, hamtouch,
    hamrepeatfromstart ] ],
[ [ my ],
  [ non_raised ],
  [ hamfist, hamextfingerl, hampalml, hamchest,
    hamclose, hamreplace, hamextfingerl, hampalml,
    hamchest, hamtouch ] ],
[ [ phone ],
  [ non_raised ],
  [ hamfinger2, hamthumboutmod, hampinky, hamextfingerl,
    hampalmdl, hamlrbeside, hamstomach, hamclose,
    hammovei ] ],
[ [ dropped ],
  [ ],
  [ ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ be_2 ],
  [ non_raised ],
  [ ] ],
[ [ busy ],
  [ non_raised ],
  [ hamparbegin, hamflathand, hamextfingerol, hampalml,
    hamplus, hamflathand, hamextfingeror, hampalmr,
    hamparend, hamparbegin, hamthumbside, hamplus,
    hampinkyside, hamparend, hamtouch, hamparbegin,
    hamreplace, hamextfingerdl, hamparend ] ],
[ [ me ],
  [ non_raised ],
  [ hamfinger2, hamthumbacrossmod, hamextfingeril, hampalmr,
    hamchest, hamtouch ] ],
[ [ punct ],
  [ ],
  [ ] ],
[ [ party ],
  [ non_raised ],
  [ hamsymmlr, hamfinger2, hamthumboutmod, hampinky,
    hamextfingeru, hampalmul, hamhead, hamlrat,
    hamtwisting, hamrepeatfromstart ] ],
[ [ have ],
  [ non_raised ],
  [ hamfinger2345, hamthumboutmod, hamextfingeruo, hampalmu,
    hamparbegin, hammoved, hamsmallmod, hamreplace,
    hamfist, hamthumbacrossmod, hamextfingero, hamparend ] ],
[ [ me ],
  [ non_raised ],

```

```
[ hamfinger2, hamthumbacrossmod, hamextfingeril, hampalmr,  
  hamchest, hantouch ] ],  
[ [ punct ],  
  [ ],  
  [ ] ]
```