Using formal logic to represent sign language phonetics in semi-automatic annotation tasks

Arturo Tlacaélel Curiel Diaz

Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le 23/11/2015 **par :**

Arturo Tlacaélel CURIEL DÍAZ

Titre:

Using formal logic to represent Sign Language phonetics in semi-automatic annotation tasks

---

JURY

ANNELIES BRAFFORT	Directrice de recherche	Rapporteur
ONNO CRASBORN	Associate professor	Rapporteur
NICHOLAS ASHER	Directeur de Recherche	Examinateur
CHRISTOPHE COLLET	Maître de conférences	Co-encadrant

École doctorale et spécialité :

MITT : Image, Information, Hypermedia

Unité de Recherche :

Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur(s) de Thèse :

Loïc BARTHE et *Christophe COLLET*

Rapporteurs :

Annelies BRAFFORT et *Onno CRASBORN*

Université Paul Sabatier

Doctoral Thesis

Using formal logic to represent Sign Language phonetics in semi-automatic annotation tasks

Author:
Arturo Tlacaélel Curiel Díaz

Supervisor:
Christophe Collet

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Équipe TCI
Institut de Recherche en Informatique de Toulouse

Monday 23rd November, 2015

"There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."

C.A.R. Hoare

UNIVERSITÉ PAUL SABATIER

Abstract

École Doctorale Mathématiques Informatique Télécommunications de Toulouse
Institut de Recherche en Informatique de Toulouse

Doctor of Philosophy

Using formal logic to represent Sign Language phonetics in semi-automatic annotation tasks

by Arturo Tlacaélel Curiel Díaz

This thesis presents a formal framework for the representation of Sign Languages (SLs), the languages of Deaf communities, in semi-automatic recognition tasks. SLs are complex visio-gestural communication systems; by using corporal gestures, signers achieve the same level of expressiveness held by sound-based languages like English or French. However, unlike these, SL morphemes correspond to complex sequences of highly specific body postures, interleaved with postural changes: during signing, signers use several parts of their body simultaneously in order to combinatorially build phonemes. This situation, paired with an extensive use of the three-dimensional space, make them difficult to represent with tools already existent in Natural Language Processing (NLP) of vocal languages.

The current work presents the development of a formal representation framework, intended to transform SL video repositories (corpus) into an intermediate representation layer, where automatic recognition algorithms can work under better conditions. The main idea is that corpora can be described with a specialized Labeled Transition System (LTS), which can then be annotated with logic formulae for its study.

A multi-modal logic was chosen as the basis of the formal language: the Propositional Dynamic Logic (PDL). This logic was originally created to specify and prove properties on computer programs. In particular, PDL uses the modal operators [a] and ¡a¿ to denote necessity and possibility, respectively. For SLs, a particular variant based on the original formalism was developed: the PDL for Sign Language (PDL_{SL}).

Finally, the development of the formal framework led to the creation of a semi-automatic annotator based on the presented theoretical principles. Broadly, the system receives an untreated corpus video, converts it automatically into a valid LTS (by way of some predefined rules), and then verifies human-created PDL_{SL}formulae over the LTS. The final product, is an automatically generated sub-lexical annotation, which can be later corrected by human annotators for their use in other areas such as linguistics.

UNIVERSITÉ PAUL SABATIER

Résumé

École Doctorale Mathématiques Informatique Télécommunications de Toulouse
Institut de Recherche en Informatique de Toulouse

Doctorat en Informatique

Utilisation d'une logique formelle pour la représentation phonetique de la Langue de Signes dans des taches d'annotation semi-automatiques

par Arturo Tlacaélel Curiel Díaz

Cet thèse présente le développement d'un framework formel pour la représentation des Langues de Signes (LS), les langages des communautés Sourdes, dans le cadre de la construction d'un système de reconnaissance automatique. Les LS sont de langues naturelles, qui utilisent des gestes et l'espace autour du signeur pour transmettre de l'information. Cela veut dire que, à différence des langues vocales, les morphèmes en LS ne correspondent pas aux séquences de sons ; ils correspondent aux séquences de postures corporelles très spécifiques, séparés par des changements tels que de mouvements. De plus, lors du discours les signeurs utilisent plusieurs parties de leurs corps (articulateurs) simultanément, ce qui est difficile à capturer avec un système de notation écrite. Cette situation difficulté leur représentation dans de taches de Traitement Automatique du Langage Naturel (TALN).

Pour ces raisons, le travail présenté dans ce document a comme objectif la construction d'une représentation abstraite de la LS ; plus précisément, le but est de pouvoir représenter des collections de vidéo LS (corpus) de manière formelle. En générale, il s'agit de construire une couche de représentation intermédiaire, permettant de faire de la reconnaissance automatique indépendamment des technologies de suivi et des corpus utilisés pour la recherche. Cette couche corresponde à un système de transition d'états (STE), spécialement crée pour représenter la nature parallèle des LS. En plus, elle peut-être annoté avec de formules logiques pour son analyse, à travers de la vérification de modèles.

Finalement, la création de cet outil à permit l'implémentation d'un système d'annotation semi-automatique, basé sur les principes théoriques du formalisme. Globalement, le système reçoit des vidéos LS et les transforme dans un STE valide. Ensuite, un module fait de la vérification formelle sur le STE, en utilisant une base de données de formules crée par un expert en LS. Les formules représentent des propriétés lexicales à chercher dans le STE. Le produit de ce processus, est une annotation qui peut être corrigé par des utilisateurs humains, et qui est utilisable dans des domaines d'études tels que la linguistique.

Acknowledgements

Before I forget, I want to express my gratitude to all of those who, directly or indirectly, contributed to the making of this dissertation.

First of all, I want to thank Conacyt (Mexico) for financing my postgraduate studies. I'm aware that I've acquired a moral debt with the mexican people and, for the entirety of my professional career, I'll try my best to repay it.

Next, I would like to thank M. Christophe COLLET, my advisor, to whom I'm truly grateful both for his support, and for putting up with my antagonistic character and my perpetually outlandish approaches. I know I was a very difficult research trainee. I was recklessly stubborn; sometimes I pushed carelessly towards very heavy stuff, without having the slightless notion of where I was, or even where I wanted to go. If this document has any structure, at all, it is because you managed to keep me from aimlessly wandering into the desert of worthless ideas that can be my mind. Thank you.

I'd also like to thank my committee, professors Annelies BRAFFORT, Onno CRASBORN and Nicholas ASHER. I appreciate your observations, your patience and, most of all, your questions, which unexpectedly gave me a better understanding of my own work. Special thanks to Prof. BRAFFORT, who took the time to painstakingly review each of my dissertation drafts, considerably lifting the quality of the final document.

My sincere thanks to Dr. Sergio RAJSBAUM, who gave me the time I needed to work on my manuscript's corrections, when I should have been working on novel solutions to distributed computing problems. I greatly appreciate your encouragement and concern these last few months, which ended up being crucial for bringing this endeavor to a happy conclusion. Unlike Mario VARGAS LLOSA, the 2010 Nobel Prize in literature, I've had more than fourteen minutes to reflect upon the significance of my own, rather minor, distinction. Now that my reflection time has passed, I can see clearly how important it was for me to have it. Thanks for your consideration.

Thanks to one of the most brilliant and passionate girls I know, my dear sister Jade CURIEL, who took time out of her schedule to read and propose changes to my linguistics chapter. I really appreciated your input. Not only were you the first person to read a portion of my thesis, but your observations heavily influenced how I presented all of the linguistic content. In addition, I ended up citing almost every reference you sent my way. Thanks for all that.

On a more personal note, I want to thank Mme. Chantal MORAND, who received me at the airport my first day in France and helped me on every step of my journey, right until

the very last minute. Merci, je n'aurais pas pu y arriver sans toute votre aide! Cette thèse vous appartient autant qu'à moi, vraiment.

I also want to thank Mme. Catherine STASIULIS, whom I met during one of my multiple clashes with bureaucracy. J'ai vraiment apprécié ton soutien, qui est arrivé pendant une période très sombre de ma vie où j'existais par inertie, isolé dans un pays étranger, sans savoir qu'est-ce que j'étais en train de faire. T'es la seule à se rendre compte que j'étais au bord du gouffre. Merci, je n'aurais pas pu m'en sortir tout seul.

Thanks to Rémi DUBOT, my research brother, my guide to french culture and, above all, a true friend. Sans doute un des meilleurs. J'ai toujours admiré ta capacité, ton élan et la force de tes principes. Quand je trouvais la thèse insupportable, tu étais toujours là pour alléger la charge; sans nos discussions, je l'aurais laissé tomber il y a longtemps. Merci. J'espère que l'on se croisera dans le futur.

My deepest gratitude to my good friend Claudio GUTIERREZ, with whom I've laughed non-stop from the moment we met. De tí aprendí a llevar las cosas más a la ligera. Me alegro que hayamos sorteado juntos el choque cultural que significaron todos estos años, el drama y la angustia de ser expatriados, de llegar a un sistema académico del cual no teníamos un claro marco de referencia. Gracias por todo el apoyo, por interesarte en mi tema de investigación y por las palabras de aliento.

I humbly thank Monia BEN MLOUKA for keeping my interest in the area alive, by providing me with French Sign Language (LSF) lessons. J'ai vraiment aimé ces séances qui m'ont gonflé l'esprit, qui m'ont donné des nouvelles idées et, surtout, qui m'ont permis de mieux te connaitre. Merci.

I can't stop mentioning Matilde GONZALEZ, the reason I came to France in the first place. Gracias por haberme integrado, por introducirme a vicios que no sabía que tenía y por los invaluables consejos que me diste sobre el funcionamiento de Christophe.

Thanks to Alain CROUZIL for stopping by to talk from time to time, even if I ended up leaving the office after dark. Surtout, merci de m'avoir aidé à surveiller les locaux la nuit.

I want to thank Anaïs MEALLIER, the first non-academic to whom I had to describe my work as a "researcher". Si cela te semblait trop ridicule c'est normal : globalement, je ne fais que de choses ridicules. Dans tout cas, j'apprécie que tu continues à t'intéresser pour ma vie, même trois ans après d'être partie. Merci.

All my gratitude to Teodora SZASZ and her wonderful mind. J'apprécie énormément que tu aies dédié autant du temps à entendre mes folies. J'en dis tellement que, maintenant, j'ai peur pour ta santé. Sinon, l'on s'est bien amusés. Moi trop, je pense. Dans tout cas, mulţumesc.

Thanks to Bérengère MATHIEU, who helped me forget the grief of leaving my old life behind by being a supportive and honest friend. Discuter avec toi est un énorme plaisir. Notre jeu sera un succès sans précédant. Merci.

I also want to thank Camille NADAL, who arrived to brighten my days after weeks of solitude, while I was in the middle of writing. Quand tu es arrivée, je commençais à péter un câble. En plus de la solitude, j'en avais tellement marre de la rédaction que mon cerveau n'arrivait plus à s'organiser. Tu m'as aidé à regagner le contrôle. Grâce à ça, j'ai pu finir le premier brouillon. Merci.

Thanks to Edna HERNANDEZ, one of the few who had to endure my worst behavior during writing, well beyond what I thought she was capable of. La neta me aguantaste más de lo que yo hubiera esperado, especialmente sabiendo que tú tambien estabas en medio de la redacción. Aunque bueno, considerando que nos conocimos en una borrachera, debería ya saber que tienes aguante. Gracias por el apoyo incondicional. No me es indiferente.

Thanks to Alexander DIAZ, Victor FUENTES and Eduardo PACHECO, who kept in touch regardless of the slight disadvantage of living on the other side of the world; an impressive feat, considering my loathing for social networks. Gracias por el apoyo, por soportar las interminables letanías que suelo escribir y por continuar incluyendome en sus vidas.

Once more, thanks to my sister Jade. Gracias por chismosearme cosas al teléfono — sobre todo los asuntos jugosos — y por hacer de IT en la casa durante mi ausencia. La verdad es que luego si me alegrabas los días.

Thanks to my parents, Arturo CURIEL and Graciela DIAZ, for giving me their unconditional support all these years, and for teaching me that hard work is more important than intelligence. A ustedes mi más grande reconocimiento. No me quedan palabras para describir lo mucho que aprecio su confianza, su apoyo. No merezco padres como ustedes. Gracias.

Thanks to the monsters from my nightmares, who keep me from standing still. I'm sure they'll eventually catch me ... until then, it is oddly reassuring to know they are there, trying.

Last but not least, thanks to Sombra the cat for taking my place at home during these long years. Now that my family loves her more than they love me, I have gained the liberty I need to do research all around the world, without ever being missed.

To every other friend and family member who has ever supported me, I extend the rest of my gratitude.

Contents

List of Figures

List of Theorems and Definitions

A mi mamá, quien más ha sacrificado por nosotros. Gracias por todo, jefa.

Chapter 1

Introduction

Sign Languages (SLs) are the visual-gestural languages of Deaf people. Broadly speaking, *signers* achieve communication through the simultaneous use of their hands and several other Non-Manual Features (NMFs), which work together to convey a message. Distinct information can be transmitted by modifying both very notorious and subtle parameters [Stokoe 1960, p. 16]; as such, changes on hand positions, shapes or movements are just as important as less salient elements like head tilting, eye gaze or body orientation [Bahan 1996, Engberg-Pedersen 2003].

This phenomenon where several body parts work simultaneously to achieve communication is called *multimodality*. Essentially, each *modality* corresponds to an individual communication *channel*, which carries the information produced by a single body part while signing. The signer parts involved on the articulation of SL utterances (hands, head, elbows, mouth, etc.) are called *articulators*. Signers receive information by observing how every articulator interacts with the others. This *simultaneous* nature of SL is further enriched with complex linguistic phenomena like *role shifting*, where signers change their point of view to "act the part" of another character during signing [Lillo-Martin 1995]. It is by way of these complex interactions that signers are capable of generating discourse with the same expressive power than vocal languages.

The use of space plays a crucial role in SL. It is through spatial relations that signers are able to expressing complex nuances on their reasoning [Emmorey 2003]. Accordingly, space has been observed to fulfill diverse linguistic roles, notably by providing *mappings* between discrete spatial regions and discourse referents [Padden 2010a]. So, for instance, a signer will commonly *map* a specific spatial location (*e.g.* by pointing) to a person, a place or an object, and will continue referring to the same entity by unambiguously acknowledging the assigned location. This phenomenon is commonly known as *spatialization*.

SLs arise naturally within Deaf communities [Senghas 2004], and provide the same expressive power as do vocal speech [Sandler 2006]. Subsequently, they serve as standalone communication systems, and exist independently from vocal languages. However, more often than not they are influenced in their structure by socio-cultural aspects related to the surrounding hearing culture [Battison 1978, Goldin-Meadow 1998].

There is no universal SL, just as there is no universal vocal language; SLs vary from country to country and from region to region. Even geographically close communities can develop independent SLs, with differing degrees of mutual intelligibility [Padden 2010b]. In addition, internal variations can be observed between signers of different generations [Woodward 1976], social classes [Parks 2010], genders [Sutton-Spence 1999b, p. 23] and ethnic origins [Solomon 2010].

It is also important to note that SLs lack of writing systems. The aforementioned simultaneity of SL parameters and their reliance on space, complicates the creation of a faithful written representation. Proposals like SignWriting [Sutton 2000], have tried to provide a written form to SLs with some success. Still, several usability considerations have undermined their utility outside of some very specific use cases, notably in research [Antinoro Pizzuto 2010, p. 211].

A more thorough introduction to SL characteristics is presented in Chapter 2 (p. 21).

The rest of this chapter presents a brief description of SL research, specially its relationship with computer science. Furthermore, the text portrays — very succinctly — the problematic at hand, and ponders about how existing tools from different domains could be used to address complex problems in the field. Lastly, the final section lists the contributions of this work, and intuitively spells out the core details of the proposed solutions.

1.1 Sign Language Research

The interest in the study of SL linguistics, has contributed towards the development of new research lines in several areas. A significant part of SL studies, are focused on deaf education [Marschark 2001] and SL acquisition [Baker 2009b]. Moreover, researchers have been able to interpret results in these fields through the lenses of neurobiology and psychology, by observing short and long term effects of SL proficiency on the human brain [Emmorey 2001].

From the point of view of education, the main interest in the area has been to assess and improve the quality of Deaf instruction. Research goes from the creation of revised pedagogic strategies in schools [Johnson 1989], to their effects on Deaf children development [Calderon 2000]. These efforts mainly strive to increase literacy rates among the Deaf, as well as to improve their integration with the surrounding hearing culture [Mayer 1996, Antia 2001].

Research in Deaf education has led to new studies on SL acquisition and its outcomes both on hearing and Deaf individuals. Work in the field has been able to document brain changes across different signing populations [Neville 1997, Nishimura 1999], which has in turn contributed to a better understanding of the human brain. In this regard, research has shown that SL learning has positive effects on the long term linguistic development of signers [Daniels 1996, Goodwyn 2000] and on their overall neurological evolution [Emmorey 2009].

This diversity of new SL studies, has called for further research on SLs themselves, especially on how their linguistic characteristics differ from those of vocal languages. For this reason, new lines have emerged on how to generate and compare meaningful linguistic data [Baker 2009a], which has led in particular to the creation of SL *corpora*.

1.1.1 Corpus-linguistics and Sign Language

In corpus-linguistics, a *corpus* (plural *corpora*) is a collection of naturally occurring language samples, intended to be representative of the language [Biber 1998]. In order to be considered representative, corpora need not only be "large enough", but samples must be selected following specific criteria depending on the purpose of its creation [Bowker 2002, p. 10]. For instance, to study use of space in SL, the collection could be based on empirical observations of spatialization occurrences. Finally, it is crucial for corpora to be representable as a digital resource, as corpus-linguistics relies on the use of computers for language analysis [Biber 1998, p. 4].

Corpora are commonly *annotated* with linguistic information [Baker 2006, p. 6], which allows researchers to derive conclusions from data. Annotations can go from the purely symbolic, to even carry meta-data about the speakers. In that sense, they can be as simple as labeling words with their syntactic function (i.e. verb, noun, etc), or as complex as noting the socio-political role of the corpus participants [Warren 2004].

Aside from annotation, corpora analysis benefits from another technique called *transcription*. Broadly speaking, a transcript intends to represent information from one medium into another [Jenks 2011, pp. 2-6]. Contrary to annotation, transcripts try to

give rigorous depictions of the original information, by applying a series of symbolic conventions [Lapadat 1999, p. 68]. For example, phonetic transcription can be done by using the *International Phonetic Alphabet*, so as to precisely convey the sounds produced by vocal utterances. Finally, it is important to emphasize that transcription seeks to create a *literal* representation of the source material and not to *translate* it, just as IPA does with vocal language sounds.

Corpus-linguistics has become a valuable resource in linguistic studies, since it readily uses new technology to discover otherwise difficult to find patterns. Current vocal corpora are usually comprehensive enough to describe a wide array of linguistic phenomena over vast amounts of information [Partington 1998, pp. 2-4]; difficulties arise, nonetheless, when working with SLs corpora.

1.1.1.1 Sign Language Corpora

Similarly to vocal languages, a SL corpus is a digital compilation of language samples documenting its use. However, because of their lack of a written representation, SL corpora are bound to be kept as video collections, whereas their vocal counterparts can usually be kept as textual records. This constraint alone is enough to complicate their creation, since it imposes the development of specialized tools for their storage, capture and analysis, among others [Johnston 2006, pp. 10–12]. These conditions hinder the growth and availability of corpora, both because of technical and financial limitations.

Commonly, different capture technologies are adopted with each new SL corpus, which has resulted on a very heterogeneous environment. Existing corpora have been created with professional video cameras [Hanke 2010], 3D sensors [Stefanov 2013], motion capture technology [Lu 2010] and even by collecting televised SL broadcasts [Forster 2014]. Moreover, each media carries its own distribution formats, which can lead to further segmentation issues between different resources. Finally, the access to financial sources such as research grants, can also have an impact on variability; for example, projects like the Dicta-sign corpus [Matthes 2012], built by several research teams across Europe, was able to produce 25 hours of high definition annotated data, which required spending on technology which was not readily available for less-financed corpora.

Differences in capture, distribution and storage formats, complicate the task of amassing big SL data repositories like the ones existing for vocal languages. For reference, the *British National Corpus*, an English language corpus, holds more than 100 million words [Leech 1992]. Likewise, the *Leipzig Corpora Collection*, which documents more than 20 vocal languages, holds about 30 thousand sentences on its worst documented language and as many as 30 million on the best documented [Biemann 2007]. However,

with SLs, even large corpora pale in comparison with the sizes of written language collections. A big SL corpus like the *DGS Corpus Project* has about 2.25 million signs [Nishio 2010], while the majority of the documented corpora hold only a few hundreds of sentences [Morrissey 2005, Bungerot 2008].

1.1.1.2 Transcription and Annotation of Sign Language Corpora

Transcription and annotation of SL corpora are highly specialized, time-consuming, endeavors [Neidle 2001, Forster 2010]. Each new corpus invariably leads to the production of machine-readable transcripts and annotations (see Subsection 1.1.1 p. 3). They are mainly used for scientific research, thus they usually can't be created by signers with no linguistics training whatsoever [Chen Pichler 2010, p. 4]. Even with simple notations tasks, SL experts are needed to prove that the produced documents are exploitable [Börstell 2014]. For this reason, it is acknowledged in the literature that there is a lack of experts capable of fulfilling the notation needs of the domain [Johnston 2006, p. 12]. Incidentally, notations are prone to human-error [Fanghella 2012, p. 59], as they tend to be very detailed; hence, it is a common practice to double check notes in order to avoid inconsistencies, which increases the quantity of work needed for the task.

Regardless of their significance, there is no clear consensus on which coding systems should be used for transcripts and annotations [Pizzuto 2001, Miller 2001, van der Hulst 2010]. Furthermore, no standard notation guidelines exist [Schembri 2010, Johnston 2011]. This situation results in variations between different notation teams, even over the same corpus [Müller de Quadros 2014, p. 97].

Transcripts of SL corpora are created by way of symbolic representation systems [Hoiting 2003], akin to those mentioned on page 2. Consequently, corpus transcription faces the same kind of problems as the development of all-purpose SL writing systems; namely, the difficulty to accurately convey every aspect of visual communication by way of a written document. Furthermore, transcripts tend to be very complex, considering the high quantity of parameters that must be represented [Johnston 2010, p. 19]. As a result, their creation is usually reserved for the study of very specific linguistic phenomena [Morgan 2003], making annotation the preferred method to represent SL information in corpora.

Most of the existing corpus annotations are *glosses* [Pizzuto 2001]. Essentially, *glossing* consists in adding a one or two word description, a *gloss note*, to a visually recognized element in the corpus. Frequently, these notes are merely words from a vocal language; however, in some cases, simple coding systems are created to better represent the elements of interest [van der Hulst 2010, pp. 29–35]. The process ensures the creation of small,

machine-readable notes, which are easier to create than transcripts, and can be used alongside corpus multimedia as needed [Johnston 2010, p. 24].

Glosses are usually created with a specific purpose, and can be of different types. Hence, a single corpus can spawn several kinds of glosses, being the most common the ones produced by the observation of lexical signs (*e.g.* dictionary signs), non-lexical signs (*e.g.* position and size classifiers), spatialization cues (*e.g.* pointing signs) and culturally influenced gestures (*e.g.* proper names) [Chen Pichler 2010, Johnston 2010]. Glosses can also be created individually for hands and NMF.

Nowadays, researchers use specialized *multimodal* annotation software like ELAN [Wittenburg 2006]. These tools permit the creation of time-aligned glosses, over corpora with more than one communication *modalities* such as speech, facial expression, hand configuration and others. They let users select a relevant time-interval, for example when a sign occurs, and mark it with a gloss note. Notes can be arranged into *tracks*, each track representing a different gloss type. Figure 1.1 shows an snapshot of ELAN during the creation process of a lexical gloss for a French Sign Language (LSF) corpus.

FIGURE 1.1: Time-aligned lexical gloss of FSL on ELAN annotation software. Gloss is written in French. [1]

More information about SL corpora and other annotation tools can be found in Section 3.3.3, Chapter 3 (p. 63).

1.1.2 Computer Science and Sign Language

The relationship between SL and Computer Science (CS) is wide, going from the development of tools for theoretical linguistics to practical applications directed towards Deaf integration. In particular, major efforts exist on the areas of SL synthesis and recognition, both of which are involved on various projects of other areas. Also, research

1. Corpus and gloss data provided by the Dicta-Sign project [Matthes 2012].

along these lines tend to overlap with linguistics on their need for corpora, which can prove troublesome due to the low availability of big SL data repositories (see Subsection 1.1.1.1, p. 4).

1.1.2.1 Sign Language Synthesis

Synthesis centers around the creation of artificial SL utterances, performed by animated human avatars [Bangham 2000, Filhol 2007, Kipp 2011]. The main objective is to communicate relevant information to Deaf users, in situations where no SL translation exists. Figure 1.2 shows the example of a signing avatar.

FIGURE 1.2: Example of an avatar used in SL synthesis. [2]

Synthesis relies on signing examples to synthesize biologically inspired movements. Research has shown that humans respond naturally to movement of biological origin [Vercher 2006, pp. 6–8]. In this regard, a more organic synthesis can effectively enhance the comprehensibility of artificial utterances [Rezzoug 2006, p. 1]. For this reason, the creation of copora for synthesis is a common occurrence, since the development of more realistic animations relies on these to learn from human signers to accurately render signing motion [Lu 2010, Lefebvre-Albaret 2013].

Synthesis deals also with SL representation issues. The movements of the avatars are usually controlled by way of technical notation languages like the Signing Gesture Markup Language (SiGML) [Elliott 2004]. Formalisms like these produce very detailed utterance descriptions, which serve as mechanical instructions for the avatar to follow. The produced descriptions are closely related to SL transcription and its multiple issues (see Subsection 1.1.1.2, p. 5). Indeed SiGML, for example, is based upon the Hamburg

2. Obtained from *3DSigner* [WebSourd 2015], commercial SL synthesis technology.

Notation System (HamNoSys) notation [Bangham 2000, pp. 4–5], only adapted to avoid describing features that cannot be animated by the target system. Figure 3.13 (see p. 53) shows the example of a sign description created in SiGML.

Other representation languages used in synthesis have gone beyond the purely geometrical into higher level descriptions. For instance, the classifier-predicates models by [Huenerfauth 2006], enables the representation of "invisible objects" around a signer. In the same vein, the Zebedee representation formalism developed by [Filhol 2009] holds similar capabilities, permitting the representation of SLs at higher levels rather than by giving thorough geometric configurations.

As with corpus annotation, the representation languages used for SL synthesis tend to be very heterogeneous due to the variety of the available resources. Additionally, each description formalism must adapt to the limits imposed by its avatar's synthesis engine, which may encompass different constraints in terms of: space, expressivity of the depicted articulators, and even synctactic rules. For example, variations on how the signing space is modeled during generation, may lead to differing representation capabilities [Braffort 2008]. Consider the case of entities located in space: the places where an entity is positioned, the possible internal relationships and any semantic modifications they may suffer, have to be expresible by the selected description formalism, which adapts to the underlying representation of space by the engine.

For this reason, existing technical notations are designed around different principles: detailed spatial representation [Marshall 2004]; accurate description of NMF [Niewiadomski 2009]; synchronization between body movements [Filhol 2012a] or for the compositional use of very low level movement descriptions [Jung 1994].

In some cases, these representations go beyond being purely mechanical. Some efforts have been made to develop *higher level* representations, capable of describing grammatical constructions rather than just geometrical animation recipes [Losson 1998, Marshall 2005]. However, work in the area is limited by a lack of consensus on how to interpret some SL linguistic phenomena [Corina 1993, Meier 2002], which ultimately shows how intertwined SL linguistics can be with areas like computer science.

Finally, a couple of all-purpose synthesis engines are readily available for commercial use [Nelson 2009], as well as several research projects focused on exploring different use cases of this technology. Among these, notable examples are the signing of recurrent information like the weather [Verlinden 2002], the creation of educational materials [Karpouzis 2007], on-the-fly translation of text to SL [Xu 1993, Boulares 2012], the distribution of internet content [Kennaway 2007] and solutions enabling translation from written language to SL [Zhao 2000].

1.1.2.2 Sign Language Recognition

Research in SL recognition is focused on the development of technology related to the automatic translation of SL into speech, written or else, and the automatic identification of SL linguistic characteristics [Cooper 2011]. It involves the creation of tools for the capture, representation and automatic analysis of SL utterances [Ong 2005].

Most of the existing work on the area is based on the use of SL corpora [Cooper 2011, p. 4]. Recognition research takes advantage of machine learning algorithms to find significant patterns on SL data. Essentially, corpora are used as a source of *training* and *test* sets for machine learning tasks; that is, corpus samples are selected to both teach an algorithm what to classify, and to evaluate how effective the classification is [Wu 1999, Ong 2005]. The idea is that, once the algorithms are developed and tested, their use can be extended to other corpora or to create practical applications.

Recognition covers a wide range of tasks going from the identification of complete sentences [Starner 1998] to the detection of phonetic properties, those measurable by electronic means. However, both technical and theoretical issues on the field have impeded the development of SL recognition technology [Morrissey 2006]; Natural Language Processing (NLP) techniques often rely on sizable amounts of data to work with [Banko 2001, Brill 2003, Halevy 2009], which is difficult to obtain with SLs. Moreover, the problem is accentuated by the lack of a written representation (see Section 1, p. 2). Consequently, the reliance on corpora arises a series of technical problems related to their format, low availability, lack of annotation and overall heterogeneity (see Subsection 1.1.1.1, p. 4).

Even at the most basic level, SLs recognition is not straightforward. Ample knowledge on the studied language and a solid linguistic theory are often needed to train recognition algorithms [Cuxac 2007a, pp. 12–14]. However, the lack of consensus regarding some SL linguistic characteristics [Corina 1993, Meier 2002] and the low availability of experts are common occurrences in field (see Subsection 1.1.1.2, p. 5). For instance, lets take the example of multimodality (see Section 1, p. 1). The interactions between the signer's body parts involved in communication are combinatorial in nature. As such, a single change in one of them can modify the meaning of the information given by the others [Brugman 2002, p. 177]. In addition, some of the combinations may be completely devoid of meaning as is the case with *co-articulations*, the arbitrary movement combinations signers execute when passing from one sign to another [Segouat 2009]. How and when body movements synchronize to create meaning is a subject of discussion in SL linguistics; however, adopting a synchronization model is fundamental even to recognize individual signs [Brugman 2002, Filhol 2012a].

The different physical characteristic of the modalities are also an issue for corpus-based recognition. Each modality usually requires individualized classification techniques to take in account the physical constraints of the signer's body. A notorious example of this are the differences between NMFs and manual features; the hands are usually very prominent and have ample degrees of articulation [Rehg 1994], whereas NMFs tend to be more difficult to follow and have to be treated with different techniques [Liu 2014, p. 673]. As a consequence, the quality of the information that can be automatically extracted from corpora will necessarily be constrained by their technical characteristics, since the technology used on their capture can favor the detection of certain features while inhibiting others [Wu 1999, pp. 4–5]. These problems impair the portability of recognition algorithms, as they imply that it won't always be possible to ensure similar recognition rates across corpora of different origins.

With this in mind, several tools have been developed with the purpose of enabling feature recognition on both recent and older corpora, regardless of any existing technical limitations [Dreuw 2008b]. For example, some projects have tried to do some analysis by calculating position information of hands and head on existing 2D corpora [Gonzalez 2012]. Others have used 3D tracking algorithms to analyze signing more reliably, by exploiting spatial information [Nickel 2003]. Motion caption approaches have also been developed, where the combined use of wearable sensors and cameras enable recognition of complex features alongside tracking [Mehdi 2002].

Regardless of the underlying technology, most research projects on SL recognition fall into two main categories: *isolated* and *continuous*. Isolated recognition attempts to single out sign appearances on corpora; that is, the system is trained with a specific set of signs, and it automatically analyzes where in the corpus these signs appear [Grobel 1997]. In contrast, continuous recognition tries to identify signs on a sequence; each signing video is analyzed and segmented into possible signs, then the system attempts to determine what sign corresponds to each segment [Liang 1998]. It is mainly continuous recognition which has to deal with complex phenomena like co-articulations, as here the system has to take in account synchronization issues to achieve segmentation [Gonzalez 2012]. In either case, the use of solid computational and linguistic models is necessary to achieve good recognition rates, specially since the number of possible combinations that could potentially convey meaning is intractable without prior language knowledge [Vogler 2001, p. 358–359].

Most of the existing work on SL recognition has centered on the identification of lexical signs [Ong 2005, p. 874], both in continuous or isolated sign recognition. Unfortunately, there exist serious size limitations in the majority of cases [Wang 2002, p. 1]. Additionally, the low scalability of corpora complicates the creation of rich sign-based recognition

models [Bauer 2002a, p. 436], which indirectly forces the training vocabularies to be small. This has lead to increased interest on raising the sizes of these vocabularies, which has gained traction over the years [Ong 2005, pp. 886–887], although with somewhat limited success.

This situation has pushed research towards sub-lexical feature recognition, launching exploratory studies on SLs morphological characteristics of signs such as motion and posture analysis [Lefebvre-Albaret 2010]; sign segmentation [Gonzalez 2012]; the study of co-articulations [Fang 2007, Segouat 2009]; hand configuration classification [Hamada 2004]; facial-movement analysis [Vogler 2007] and on the synchronization between manual and non-manual features [Ma 2000, Vogler 2001]. Sub-lexical research has also benefited from the introduction of new, cheaper, capture technology [Zafrulla 2011, p. 280], which has increased the quantity of exploitable information on new recognition corpora.

Finally, research above the lexical level has mainly focused on the detection of complete sentences [Starner 1998, Zahedi 2006], sign alignment between different SLs [Chiu 2007] and recognition of some spatiotemporal features [Kelly 2009]. However, little-to-no work has been directed towards SL discourse translation. This is because translation relies on both lexical recognition and on correctly interpreting the complex physical interactions of articulators (see Section 1, p. 1), non-trivial tasks themselves [Cuxac 2007a]. Nonetheless, some discourse analysis models have been developed [Braffort 2006, Dalle 2006] in an effort to at least determine which is the minimum level of automatic analysis needed, to develop a discourse translator.

1.2 The Problem of Semi-Automatic Annotation

Annotation plays an important role in computer-aided SL research; it is by way of annotation that automatic systems can discriminate relevant information from corpora, to be used in research [Dreuw 2008a, p. 1]. By reading annotations, systems become capable of removing noisy video-frames automatically, effectively creating *ground truths* for machine learning algorithms [Moehrmann 2012]. However, the annotation process is very time consuming and requires the participation of human experts, since only them are capable of judging which features are relevant enough to be annotated. Nevertheless, regardless of expert availability the task may appear endless, since new linguistic hypotheses may diminish (or increase) the perceived importance of notable features, thus leading to new annotation demands [Brugman 2002]. This implicitly points towards a constant annotation cycle, were experts elaborate over previous iterations. Moreover, due to the multimodal nature of SLs, needing different types of gloss is a common occurrence (see Subsection 1.1.1.2, p. 5), increasing even more the dimensions of the task.

Different research projects entail different annotation needs. In SL synthesis, for instance, glosses can be used to manually indicate where the relevant movements to be learned are [Arikan 2003, Chao 2004]. For example, a system trying to mimic natural eyebrow movement, will most likely require a gloss marking every interval where the signer is actively using his eyebrows in a meaningful manner.

SL recognition, more than synthesis, relies on annotated video-data. It is by way of linguistic annotation data that an algorithm collects enough information to recognize individual signs [Nickel 2003, Ding 2009], relevant facial features [Von Agris 2008, p. 3] and signing or non-signing segmentation patterns [Han 2009, Yang 2009]. In this regard, annotations are used to make sense of data in automatic tasks and, as such, serve as ground truths for evaluating how effective each task can be [Moehrmann 2013]. Additionally, recognition techniques can serve as support for many areas; in synthesis, for example, recognition algorithms can be used to find grammatically significant movements [Erdem 2002, Neidle 2009], which could later lead to better quality synthesis.

Despite of the importance of annotations for SL research, their availability is limited. The ever present need for linguistic experts, the high specialization of annotation tasks (see Subsection 1.1.1.2, p. 5) and the heterogeneity of corpora (see Subsection 1.1.1.1, p. 4), hampers the production of new, correct annotations. The problematic further deepens if we consider that new linguistic knowledge can weight on the adequacy of previous and future annotations [Schembri 2010, p. 3], which points to the need of constant annotation cycles over the same corpus. This situation has prompted research on the automation of at least part of the annotation process, in an effort to streamline the creation of new annotations regardless of corpora [Neidle 2001, Braffort 2004, Dreuw 2008a].

Different automatic and semi-automatic annotation projects have emerged, with applications in SL research. Research on annotation of image datasets covers the detection of relevant gestures in conversation [Wilson 1996], the identification of human actions [Duchenne 2009] and annotation of body language [Chippendale 2006]. Other efforts more directly related to SL come in the form of complete platforms with automatic annotation algorithm integration [Neidle 2001, Braffort 2004], the development of automation modules for already existing annotation tools [Dreuw 2008a] or the creation of corpus-specific software, aimed to ease annotation and transcription of distinct corpora [Yang 2006, Pitsikalis 2011, Gonzalez 2012].

Unfortunately, automatic support for annotation relies on the same techniques used for SL recognition. Consequently, researchers have to deal with the same kind of problems present in the aforementioned area; namely the lack of homogeneous corpus data, shaky theoretical linguistic frameworks and the absence of a written representation [Morrissey 2006]. Moreover, this relationship between the two areas implies that the development of new

annotation algorithms depends on the existence of previous annotations from which to learn from. Consequently, it's not difficult to see that, being a recognition task in itself, the creation of an automatic annotation algorithms will only be possible for existing annotation schemes where enough data is available; annotations based on new linguistic hypotheses, and cases with annotation few examples, will still be difficult to create.

In short, automatic annotation is limited by:

— The lack of a written representation, which contributes to the scarcity of large data repositories, as the ones needed for the application of classical NLP techniques;
— the heterogeneity of corpus-data, which complicates the reutilization of automatic analysis algorithms across different corpora;
— the changing nature of annotation tasks, very theory-dependent, which hampers the development of standard annotation templates.

In this regard, the work presented in this thesis is geared towards the representation of SL in semi-automatic annotation tasks, as introduced in the next section.

1.3 Contributions

This thesis, presents the application of *formal methods* for the description of SL in computer-based research. Roughly, formal methods are mathematically-based techniques used to rigorously describe (*specify*) complex systems and their properties. They provide theoretical tools to build, develop and validate systems regardless of their complexity, maximizing confidence and minimizing unexpected behaviors [Monin 2003]. To this end, formal methods often rely on *formal specification languages* to specify system processes. A description created with a specification language is called a *formal specification*, and they serve as guidelines on how a system should work. One of their main advantages is that they can be mathematically *verified*. That is, it is possible for an automatic algorithm to determine whether the described system satisfies certain properties or not, ensuring that the described process is error-free. The driving idea behind this document's contributions, is that the same techniques can be used to specify SL productions and verify their characteristics, which in turn can provide some insight into the development of automatic annotation systems.

Several industries rely on the use of formal methods for the implementation of their industrial processes. For instance, formal verification is commonly used in the development of safety-critical systems [Bowen 1993], communication technology [Barbulescu 2004], cloud computing [Boer 2012] and complex software systems [Woodcock 2009]. Finally, formal methods are not limited to industrial uses; the same tools have applications

for research in biology [Hartel 2005, Zambach 2010], group interaction [Hoek 2003] and linguistics [Kamp 1993, Moss 2007].

This thesis presents the utilization of formal methods to transform corpora into an intermediate representation layer, where automatic annotation algorithms can work in better conditions than when working directly with video sequences. The main idea is that corpora can be described with a formal specification language, instead of a classical SL transcription language. Moreover, we prove that the proposed formal language is more expressive than any of the existing SL notations, and that it can be adapted to use the same symbolic conventions.

By describing each corpus with a single formal specification language, heterogeneity becomes less of a problem. Incidentally, this can increase the sizes of the available datasets. Moreover, any algorithm developed to work with these formal representations of corpora can be reused on any new corpus, given that it can be described with the same language. In essence, this method permits seeing each corpus as a standalone system with well-defined rules. The added advantage is that annotations can be conceived as observable properties of the system and, as such, can be found by way of formal verification. A diagram depicting the process is presented in Figure 1.3.

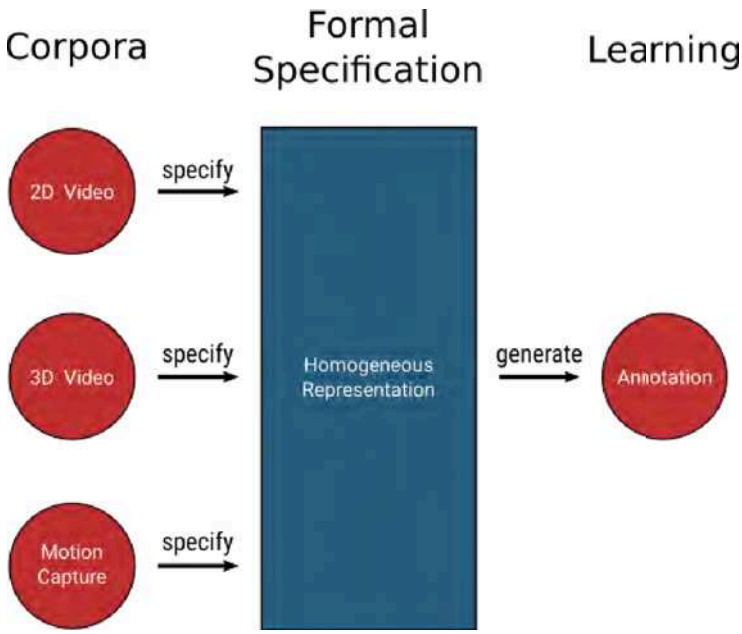


FIGURE 1.3: Diagram depicting the proposed intermediate layer between corpora and learning tasks. Each corpus can be specified into an homogeneous representation, that can then be used to verify the existence of the required annotations.

More information on formal methods and their applications can be found on Section 3.4, Chapter 3 (p. 67)

1.3.1 The Propositional Dynamic Logic for Sign Language

A modal logic was chosen as a formal specification language based for corpora: the Propositional Dynamic Logic (PDL). This logic was originally created to specify and prove properties on computer programs [Fischer 1979]. PDL uses the *modal operators* $[a]$ and $\langle a \rangle$ to denote *necessity* and *possibility*, respectively. For instance, the expression $[a]p$ has the meaning that, after performing action a, p will *necessarily* hold. On the other hand, the expression $\langle a \rangle p$ describes the situation where, after executing action a, p may *possibly* hold.

In general, PDL permits the encoding of *change* by way of formulae. For instance, the formula $f_1 \rightarrow [a]f_2$ can be read as "If f_1 holds, then f_2 will necessarily hold if action a is executed". When f_1 is a proposition meaning "my bucket has water", a is the action "I empty my bucket" and f_2 is the negation of f_1, then the formula reads "if my bucket has water, then necessarily my bucket won't have water if I empty my bucket". The same idea can be applied to SL articulators, so as to model them in terms of their actions. For example, if f_1 is a proposition meaning "the right hand is touching the head", action a is the action "the right hand moves to the chest" and, once again, f_2 is the negation of f_1, then the formula would read "if the right hand is touching the head, then it won't be touching the head if it moves to the chest".

For SLs, a particular variant of the PDL was developed, the Propositional Dynamic Logic for Sign Language (PDL_{SL}). With the PDL_{SL}, articulators are interpreted as independent agents, where each articulator has its own set of valid actions and propositions, and executes them without influence from the others. It is important to note that this does not reflect the relationship between articulators in linguistics: in real life, articulators are often dependent on each other, for various reasons. However, it is important for PDL_{SL} to be able to establish these kind of relationships explicitly rather than implicitly, as this contributes to the flexibility of the language.

The simultaneous execution of different actions by several articulators yield distinct situations, which can be modeled as graphs for formulae interpretation. In essence, these graphs depict time intervals without change as vertices, while the edges are taken to represent intervals where changes occurred. Returning to articulator independence, it is the graph where any dependent articulatory actions clearly show, as they will always occur simultaneously, on the same edges. A more detailed explanation of this interpretation can be found on Chapter 4, p. 77.

The use of a formal logic as specification language brings flexibility to the task, since it is possible to encode any information by formulae as long as it can be thought of as a propositions (*i.e.* a true-false statements). This adaptability, alongside the existence

of well established inference and verification methods, makes it a good choice to deal with situations where incomplete or erroneous information is present; by using logic, it is possible to prove automatically and without ambiguity whether a certain specification is inconsistent and correct it. Additionally, the inherent *compositionality* of formulae makes it possible to mix them, separate them and reuse them to create new expressions, which comes useful to specify multimodality (see section 1 p. 1).

Finally, the use of logic not only benefits corpus specification, but enables the creation of *annotation templates*. These are none other than PDL_{SL} formulae describing the desired properties to be found on corpora. So, for instance, if the formula $f_1 \rightarrow [a]f_2$ seen above where to be verified on a corpus (with f_1 being that the right hand touches the head, $f_1 \equiv \neg f_2$ and a meaning that the right hand moves to the chest), the result would be every state where the right hand started touching the head and ended over the chest. This information can then be poured into an annotation tool, in order for a human expert corroborate the results and correct the annotation if needed, simplifying the overall process.

The complete definition of PDL_{SL} is given on Chapter 4, p. 77.

1.3.2 Semi-automatic Annotation using PDL_{SL}

The choice of PDL_{SL}, led to the creation of a framework for the implementation of semi-automatic annotators based on this logic. The framework was designed to follow the architecture presented in Figure 1.4. The idea is that the system receives an untreated corpus, converts it into a graph respecting the definitions given by the PDL_{SL} and uses it to generate an editable annotation file. Several modules along the way let the users determine how the corpus will be analyzed, what kind of annotations will it generate and how the graph will be created.

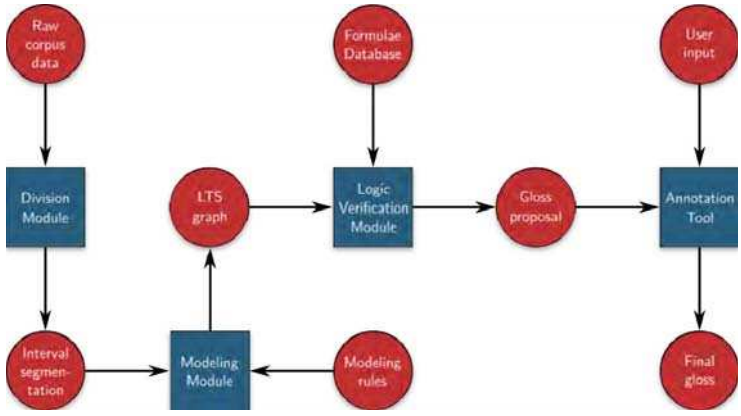


FIGURE 1.4: Block diagram of a semi-automatic SL annotator based on PDL_{SL}.

Roughly, the Division Module uses a feature tracker to follow the articulators of interest and determine if there were any changes at any given time. The implementation developed for this thesis, uses a head and hands tracking algorithm [Gonzalez 2011] over Dicta-Sign corpus [Matthes 2012]. The tracker reports 2D positions of hands and head on a frame-by-frame basis, which are used to segment each video into two types of intervals: movement (*change*) intervals and static intervals. The Division Module generates an independent interval sequence for each articulator as follows: each movement interval is selected by determining if the speed of movement of the articulator of interest is above a certain threshold [Gonzalez 2011]. Static intervals, as their name imply, are intervals where the speed of the articulator of interest is essentially zero.

The set of interval sequences is passed to the *Modeling Module*, who is in charge of pouring the tracker information into a formal specification. For this, it needs a group of "Model rules", which are just the set of PDL_{SL} propositions and actions defined for the selected corpus. Each of this propositions and actions is linked to a decision method; a function used to decide if they are true or false at any given time. Broadly, the Modeling module takes each of the static intervals generated by the Division Module, and uses the decision procedures to determine which propositions happen to be true within the interval limits. Each static interval will then be represented as a state, labeled with all the found true-valued propositions. Once the states are calculated, each movement interval separating two static intervals is converted into an edge connecting the two corresponding states. The edges are labeled as-well, only this time the module tests actions rather than propositions. The final result is a set of per-articulator Labeled Transition System (LTS), which can be merged into a single graph with the procedure presented in Section 4.3 (see p. 89).

The merged graph is passed to the *Logic verification module*, where an automatic algorithm performs the annotation procedure. The module receives a "Formula Database", which contains a series of formulae created by a human annotator. Each formula encodes the description of a desired annotation as a property, so that the module can use it as input for a logic verification algorithm (see Subsection 3.4.1, p. 68). The verification procedure is straightforward; for each state on the graph, the module will verify if each formula on the database is satisfied. If it is, it will add it to the list of proposed annotations, alongside the corresponding time-frame where the annotation was found. If a formula is not satisfied, it is simply discarded.

Finally, the list with the proposed annotations is converted to the format of a multimodal annotation tool. Here, an expert can change the automatic results to correct mistakes or to add more information to the final gloss.

A more detailed presentation of the annotation framework and the implemented algorithms can be found in Chapter 5, p. 99.

1.3.3 Empirical Application of PDL_{SL} to Sign Language Annotation

The final contribution of this work, consists on the analysis of the use of the proposed annotation framework on real world data. For this, a set of properties were encoded as PDL_{SL} formulae and were later given to an automatic annotator, implemented with the help of the annotation framework previously presented. The process was used to find properties over videos from the Dicta-sign corpus [Matthes 2012]. An example of a resulting gloss can be seen on figure 1.5. In the image, a user defined the property `BOTH` as `RIGHT` $\wedge$ `LEFT`; that is, the system has to find every time interval where both the right and left hands are static (marked as `KEYPOSTURE` on the figure).

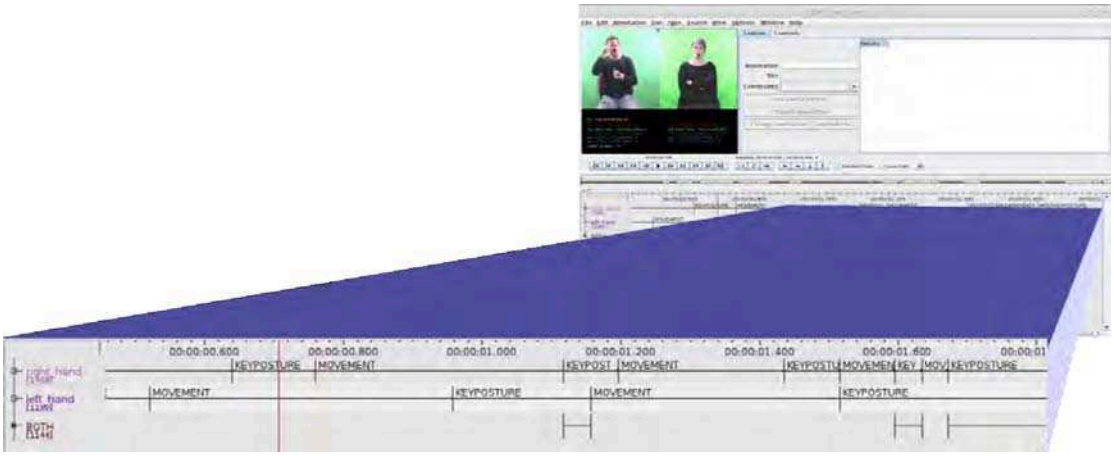


FIGURE 1.5: The resulting annotation of property `BOTH` on ELAN annotation tool. The intervals where `BOTH` was found, are shown on the third annotation track from up to bottom.

The obtained results and their qualitative analysis are detailed on Chapter 6, p. 129.

1.4 Conclusions

In brief, this thesis presents a series of problems related to the semi-automatic annotation of SL corpora and a method of solving them by way of a formal logic:

— The lack of a written representation for SL corpora was addressed by the use of a formal specification language, the PDL_{SL}, which let annotation algorithms work with an intermediate corpus-representation rather than directly with images;

— the heterogeneity of corpus-data was acknowledged by the creation of an automatic annotation framework, designed to create graph representations from any corpus regardless of the technology used for its creation; and, finally,
— the changing nature of annotation tasks was faced by using PDL_{SL} formulae as annotation templates, which can be easily changed and reused to adapt to different needs.

A more thorough analysis of this work's results can be read in Chapter 7, p. 147.

Chapter 2

Sign Language Overview

Sign Languages (SLs) are complete, complex standalone visual languages. They are often independent from vocal communication systems [Emmorey 2001] and they are not *universal*. In particular, they are as varied as vocal languages, have the same expressive power than these, and possess their own particular grammars and vocabularies [Wilbers 1987]. There do exist artificial SLs like the International Sign Language (ISL) trying to improve global Deaf communication. In essence, ISL follows the same philosophy than Esperanto [Tonkin 1997], in the sense that it tries to group common grammatical structures and signs into a single language to facilitate learning [McKee 2002]. Nevertheless, even in artificial cases like this one, lexical and syntactical variations are present, contributing to diversification [Beckner 2009, p. 15].

SLs can be found across most countries, in most cases with their own particular rules and linguistic characteristics. As such, British Sign Language (BSL) and American Sign Language (ASL) are completely different languages, even with low mutual intelligibility [McKee 2000]. Here, it is important to stress that SLs are not visual forms of vocal languages; as such, French Sign Language (LSF) and ASL, are completely different languages from French or English [Emmorey 2001]. This doesn't mean that SLs are not influenced by vocal languages, but rather than their grammatical structures differ; they are not versions of the same language [Sutton-Spence 1999a].

Even though SLs are complete languages, they differ from speech in some aspects. Mainly:

— **They are visual:** SLs are based on gestural expressions, characterized by movement, rather than by sounds or written symbols.
— **They have multimodal structures:** they convey messages by way of complex interactions between the hands and other parts of the body.

It is important to note that there is a difference between *multimodality* as used in SL linguistics, and how the term is understood elsewhere. In vocal linguistics, multimodality refers to the use of different human communication channels at multiple levels of abstraction, which together are able to modify the meaning of a message [Gibbon 2009, p. 11]. As such, variations such as voice intonation could modify or reinforce what an utterance ultimately represents. In contrast, SL modalities contribute specific information to the task of building meaning; in consequence, each of the different communication channels has a predefined role, crucial from the start for the formation of even the smallest meaning units [Cuxac 2007b, pp. 14–15].

— **They make extensive use of space:** spatial locations around the signers fill several grammatical roles and are fundamental to build discourse.

— **They depict *iconicity*:** signers depict with their body different levels iconic representation, going from mimicking characteristics of real-life objects, to assuming the role of an external entity during discourse [Cuxac 2007b].

The characterization of SL corporal expressions, has been an object of study in linguistics since the end of the XIX century; however, few linguistic studies exist, comparing the properties of SLs with respect to vocal languages. Regardless, some well established theories have emerged in the last few decades, geared towards gaining a better understanding of the underlying mechanisms ruling gestural communication.

In the following paragraphs, a brief cultural review is presented, outlining the evolution of SL and their study in western cultures. Following this short outline, a more thorough introduction to SL linguistics is presented, which extends until the end of this chapter.

2.1 Cultural Context

For the first decades of the XX century, SL research had been directed towards the acceptance of SLs as full-fledged natural languages; only in the last two decades, they have started gaining the same status as vocal languages around the world (*e.g.* as recently as 2005 for the European Union [Timmermans 2005]). Before that, SLs were hardly ever considered part of human language, and concepts such as regional denominations (like LSF or ASL) were not even heard of [Slobin 2006]. This constant struggle towards SL acceptance has had important effects, not only on how they are studied, but also on how the language themselves evolve [Frishberg 1975].

The earliest recorded mention of the Deaf and the existence of gestural languages, comes from Hellenic Greece. Both Plato and Aristotle made clear references of the use of

SLs by their contemporaries [Cerney 2004]. However, the conception of the Deaf as a community united by language, or even just the use of SL in education, would not become a common practice until the XVI century [Baynton 1995]. Before that time, deafness was understood by following the initial hypothesis of Galen (the Greek physician), who thought that hearing was affected by speech and vice-versa. Therefore, integration was centered around forcing the Deaf to speak, assuming that it would improve their hearing [Eriksson 1998].

With time, religion played an important role on changing the views of the people on Deafness, promoting new forms of integration for the Deaf into society. For instance, the Deaf were allowed to participate ceremonies of the Hebrew tradition, even though it prohibited most people with disabilities from entering the temple. On medieval times, Christians worked with Deaf communities while trying to spread the Gospel among them [Tebbe-Grossman 2008]. Finally, around the XV century, Deaf education in Europe started to be completely overtook by catholic monks, in an effort to introduce them into confession [Stokoe 1960]. As such, it was inside the monasteries that strong SL communities started to blossom.

The modern linguistic study of SLs can be traced back to France in 1750 [Cerney 2004]. On that year, Abbe Charles-Michel de l'Épée undertook the teaching of two deaf-mute sisters. From that moment on, and for nearly three decades, he directed an education center based on SL, open for every child born Deaf in France. His observations and theories where published on a study called " *L'institution des sourds et muets, par la voie des signes methodiques*" [L'Epée 1776], where he documents his personal findings [Stokoe 1960]. The subsequent evolution of SLs both in France and the rest of Europe, was taken over by generations of Deaf children who grew up on l'Épée's methods, only to become teachers themselves, to other Deaf children. However, much of the original practices based on deafness misconceptions have remained until these days.

Despite the popularization of SL-based instruction, speech-based methods (also called "oralism") has kept on being a central part of Deaf education. Oralism consists on teaching Deaf children to use finger-spelling, vocal speech and lip-reading to communicate, in detriment of SL [Enerstvedt 1996]. More specifically, oralist methods teach students how to spell vocal words with finger signs, forcing them to retain vocal language syntax [Sperling 1986] and overlooking SLs in favor of speech therapy. This school of thought is still prominent in several places around the world, since its supporters argue that it encourages integration [Hogan 2008]. However, the use of speech-based learning continues to be a contentious issue in Deaf education, to this day [Eriks-Brophy 2004].

In the next section, a brief introduction to SL characteristics is given. Simultaneously, some insight on how SLs linguistics are related to vocal language linguistics is presented, alongside some general notions of SL use.

2.2 Sign Language Linguistics

Linguistic research of SLs has gained new dimensions beyond the search for legitimacy, striving towards gaining knowledge about the languages themselves. To this end, linguists have adopted the basic hierarchical structure commonly used in vocal language linguistics [Akmajian 2010], dividing language studies into hierarchical *levels of analysis* such like:

1. **Phonology**.- Studies the smallest units of sound in a language, called *phonemes*, and the implicit system of rules determining how these are produced by a speaker. It is deeply related to *phonetics*, the science which studies how speech sounds are produced, transmitted and perceived by a speaker [Hayes 2008]. Thus, unlike phonology phonetics deals with real, measurable data.
2. **Morphology**.- Describes how the smallest units of meaning, *morphemes*, are combined into *lemmas*[1], or *lexical units*. In essence, it describes how words are formed [Lieber 2010].
3. **Syntax**.- Explores how the composition of lexical units form sentences.

Complementary levels and alternate classifications can be found in the literature, however these three have long been of special interest for SL studies [Kegl 1976].

The aforementioned levels of description tend to build upwards from very small structures. As such, for the rest of this document, aspects related to phonetics will be referred to as "low level" whereas more abstract concepts such as morphology or syntax will be identified as "higher" levels.

2.2.1 Phonology and Phonetics

As explained in previous paragraphs, phonology is mainly concerned about language sounds and how they are combined by humans for communication. More generally, it may be understood as the basis for linguistic study, since it aims to describe the smallest language units well before the formation of meaning. However, when talking about SLs, phonology is not properly related to sound, but rather to the movements and forms

1. Not to be confused with the term *lemma* as used in mathematics, where it is usually characterized as a "helping" or "minor" theorem with respect to a more powerful result. Further appearances of the term will refer to the mathematical convention, rather than the linguistics one.

that human signers adopt while signing. In this manner, SL phonology rests on the assumption that signs are constructed upon meaningless building blocks, akin to vocal phonemes [Axelsson 2002, pp. 14–16].

By extension, SL phonetics centers around the physical elements intervening in the formation of these soundless phonemes: this is, where classical phonetics cares about productions of the vocal apparatus, its SL counterpart examines expression, posture, and other measurable visual aspects during *articulation*.

In vocal linguistics, articulation refers to the act of generating language sounds by affecting specific *places of articulation*[2] with *speech organs* such as lips, tongue or other. Each of these, may have one or various places of articulation, which are areas such like the teeth or palate. Speech organs, also called *articulators*, may act as a places of articulation themselves and vice-versa, depending on their role during the formation of a phoneme. In essence, when an articulator affects a place of articulation, it changes the characteristics of the sound produced by the vocal cords. This enables speakers to generate a range of different *phonemes*, which are the minimal language noises forming utterances. Likewise, SL users articulate by affecting various places of articulation, in order to display phonetic configurations; the main difference, is that — instead of describing sounds — SL phonemes differ visually. As a result, it becomes apparent that SL phonemes are not articulated with the same speech organs used to generate vocal languages.

SLs are traditionally described as being mainly articulated by way of the hands, in concert with a set of other, non-manual, articulators. These visual organs affect places of articulation outside of the vocal tract, such as body and spatial regions distributed around the signer's upper body. In some cases, signers will even articulate on empty spatial locations, over places not directly touching their bodies. These fairly radical changes in language production, with respect to vocal languages, have lead to a tad of different theories trying to determine which visual cues are relevant for the phoneme formation process, and which are not [Axelsson 2002, pp. 14–16].

Phonemes, both in SL and vocal languages, are highly combinatoric in nature: the same set of units can create distinct meanings when produced in a particular order. Also, they differentiate from one another by changes on their *phonetic features*. It is important to note that, in vocal languages, each phoneme may correspond not only to a single sound, but to a set of different sounds; for instance, the sound related to the English vowel [a] may change between native speakers from different regions, whom may recognize the same phoneme even when spoken with different pronunciations. In this

2. Also *points of articulation*.

example, even though the sounds are not the same, they are still identified as being the same vowel. SL phonemes have been observed to follow the same principles, where each phoneme may have different *realizations* [Valli 2000, pp. 256 –257], based on how several visual phonetic features are modified. In later chapters, a more detailed overview of how phonetic features of SL are represented, since the main contribution of this thesis is devoted to provide new formal tools for SL phonetic representation. Nonetheless, a brief introduction to SL phoneme formation is given in the following subsections, starting with a brief description of the hands' functions as SL articulators, and their capabilities to display phonetic variations.

2.2.1.1 Hands

In SL, the hands are the most productive articulators. They are widely displayed during signing, and their flexibility and movement range let them articulate a vast number of different combinations. In essence, they roughly correspond to the definition of *active articulators* in articulatory phonetics of vocal languages [Ladefoged 1996].

For their study, hands are distinguished as either *dominant* or *non-dominant*[3]. The dominant hand, is the hand that plays the *active* role during signing; this is, it is the one preferred to perform motor tasks. In contrast, the non-dominant hand plays a *passive* role during signing; in some cases, it will even remain static. Generally, the signer will use his or her dominant hand to execute larger, more complex motions, whereas the non-dominant hand will usually be relegated to perform less salient movements [Battison 1974]. The choice of right-dominant or left-dominant can vary from signer to signer; however, the roles' functions remain the same, regardless of which hand is playing which.

With respect to the use of the hands, signs can be classified into three categories:

— **Single-hand signs**.- These are signs where only one of the hands articulate, either the dominant or the non-dominant hand.
— **Two-hand signs**.- In these, both hands articulate simultaneously.
— **Buoys**.- This is a special case where one of the hands articulate (usually the dominant) while the other one remains fixed over some spatial location. The static hand serves as a deictic marker [Lyons 1977], usually referring towards a discourse entity or directly acting as the representation of one.

Hands can affect several articulation places around a signer's body, with the head, neck, torso and arms being the most common. Hands can also, and often do, serve as points of articulation for two-handed signs. For instance, in Figure 2.1, the signer articulates

3. Can be also referred to as the *weak hand*.

the right hand over the left hand, which remains fixed during the entire production of the sign.

FIGURE 2.1: In sign $\texttt{DATE}_{\texttt{LSF}}$ the right hand affects the left hand. [4]

It is important to note that this is not, by any means, an exhaustive list; each SL will use its own distinct set of articulation places, which can differ in number and position from all the others. Figure 2.2 shows a rough diagram with these and other possible articulation places.

FIGURE 2.2: Common body articulation places. [5]

The same phenomenon can be seen with vocal languages: they produce their own distinct set of sounds, depending on how speakers affect specific articulation places. Broadly, these would correspond to *passive articulators* in vocal languages articulatory phonetics [Ladefoged 1996].

It is not necessary for the signer to have physical contact with the body for the hands to articulate; they may hover around the approximate region, as long as the reference to a point of articulation is clear, as is the case with deixis in vocal languages [Lyons 1977].

4. Images obtained from [Sématos 2013].
5. Corpus image obtained from Dicta-Sign project [Matthes 2012].

As previously stated, there are also instances where the points of articulation are not bound to body parts. Some articulations take place on the *neutral space*, an unmarked physical space in front of the signer's chest, where signs with no fixed body location are produced. In these cases, the articulation place is not linked to the signer's body but to the space itself.

2.2.1.2 Other articulators

Besides the hands, other body parts can provide active articulation, such as the head or the mouth [Crasborn 2008b]. It is common for some articulation places to change between being active or passive during signing. For instance, the head is used as the articulation place for signs like $\texttt{EAT}_{\texttt{LSF}}$ or $\texttt{THINK}_{\texttt{LSF}}$, as shown in Figure 2.3.

$\texttt{EAT}_{\texttt{LSF}}$ $\texttt{THINK}_{\texttt{LSF}}$

FIGURE 2.3: Signer articulating single-handed signs over head-located places of articulation. [6]

However, the head will sometimes serve as articulator. For instance, in the sign $\texttt{SLEEP}_{\texttt{LSF}}$, the signer articulates with his head by tilting it towards the open palm, as shown on Figure 2.4.

In this sign, it is the palm who acts as the head's point of articulation. Other body parts such as the eyebrows, lips and cheeks hold similar characteristics, as they are also capable of articulation [Frawley 2006].

2.2.1.3 Multimodality

The flexible nature of SL articulators, gives origin to the phenomenon of *multimodality*: multiple articulators acting synchronously to build language units together [Sandler 1989]. As briefly explained at the beginning of this chapter, multimodality in the context of

6. These and subsequent sign images were obtained from [Sématos 2013].

$\texttt{SLEEP}_{\texttt{LSF}}$

FIGURE 2.4: Signer articulating the sign $\texttt{SLEEP}_{\texttt{LSF}}$ with his head, using the palm of the dominant hand as place of articulation.

vocal languages has a different meaning than the one used to study SLs. In the former, multimodal information seems to *modulate* speech by giving, for example, iconic cues on different communication *channels* (voice, gesture, etc.), in order to better convey the semantics of a message [Vigliocco 2014]. In SL, however, multimodal information is crucial to understand any message, since very specific data is provided by each channel during the sign formation process[7]. In this sense, signers need to receive as much information as possible from all modalities, in order to render utterances understandable. In other words, at any given time, several body parts such as the hands, head, lips and others, will articulate simultaneously in order to build a phoneme[8]. Just as with vocal languages, phonemes are rendered distinguishable by combining different articulation parameters in a unique way.

Informally, SL modalities can be interpreted as communication channels carrying the information generated by each articulator (manual or otherwise). Under this definition, the "right hand modality" makes reference to the channel that holds information produced by the right hand.[9] The information transmitted by each individual channel will be necessarily unique, since articulators are morphologically different from each other and play very specific roles in utterance production.

2.2.1.4 The Movement-Hold Model

Signers manage to arrange their articulations into well defined *time-units*. A time-unit, is none other than the interval of time in which a phoneme takes place. The morphological

7. SL linguists like [Cuxac 2000], use a specific term for this phenomemon: *multilinearity*.

8. Some authors call the basic units of SLs *cheremes*, but this document will use the term *phoneme* for clarity.

9. The terms *channel* and *modality* will be used interchangeably for simplicity; nevertheless, the reader should be aware that the literature might define the same concepts under different names.

information contained inside a time-unit is called a *posture*. Broadly speaking, time-units are what constitutes phonemes in SLs; hence, their postures need to be unique enough to make them distinguishable from other phonemes.

Similar to vocal languages, SL linguistics study how groups of phonemes build up to create meaning. Current sign-formation theories, adhere to the idea that the sequential structure of time-units is phonologically relevant; in this view, SL signs are formed by well-defined configuration sequences, which together convey meaning.

The first description framework developed under these assumptions was the Movement-Hold Model (MHM), proposed by [Liddell 1989]. The MHM interprets time-units as utterance *segments*, and divides them into two distinct groups: *Holds* and *Movements*. In essence, a Hold is a segment with a fixed posture; this is, its information won't change throughout the entire time-unit's duration. On the other hand, Movements are segments where changes do occur. Usually, Holds will be connected to Movements in the phonological sequence. In a way, Movements can be seen as transition segments between two postures, as shown in the diagram presented by Figure 2.5.

Hold (H_A)	Movement (M_{AB})		Hold (H_B)
Posture A	Posture A	Posture B	Posture B

FIGURE 2.5: Representation of a H-M-H sequence in the MHM. Holds H_A and H_B are characterized by postures A and B, respectively. The movement between holds is a transition from postures A to B.

Signs formed by two Holds, separated by one Movement, are said to have a H-M-H structure. As previously stated, each segment is characterized by its corresponding posture (or postures, in the case of Movements). A more concrete example is shown on Figure 2.6, where the H-M-H sign $\texttt{ROAD}_{\texttt{LSF}}$ is phonetically decomposed.

H-M-H is not the only existing sequential structure; structures like M-H, M-H-M-H and M-M-M-H are also common in languages close to ASL. However, these may vary between SLs.

Recently, the authors have proposed a new model based on the MHM, known as the Posture-Detention-Transition-Steady Shift (PDTSS) system [Johnson 2011]. The PDTSS also treats signs as sequences of Holds and Movements of sorts, although with a more refined segmental subdivision.

The main segment types in the PDTSS system are Postures and Detentions, broadly defined as follows:

Postures (P) These are time intervals with associated hand configuration and location features.

Hold (H_A)	Movement (M_{AB})	Hold (H_B)
Right configuration BEAK	–	Right configuration BEAK
Left configuration BEAK	–	Left configuration BEAK
Right orientation OUTWARDS	–	Right orientation OUTWARDS
Left orientation OUTWARDS	–	Left orientation OUTWARDS
No NMFs	–	No NMFs
Right location FRONTFACE	Move towards SIDEFACE 1	Right location SIDEFACE 1
Left location FRONTFACE	Move towards SIDEFACE 2	Left location SIDEFACE 2

FIGURE 2.6: Information contained in each time-unit for the H-M-H sign $\texttt{ROAD}_{\texttt{LSF}}$.

Detentions (D) Correspond to Postures where hands have no movement.

Together, these classes are roughly equivalent to Holds in the MHM model. In a similar fashion, Movements are also divided into two classes, as show below:

Transitions (T) These are time intervals attached to hand changes between two Postures, similar to the Movement shown in figure 2.6.

Steady Shifts (S) Roughly, they are like Transitions but executed in a slow, steady movement.

Unlike their previous work, the authors' new model does not allow the existence of two contiguous Movements without a Hold separating them: sign structures are mapped to strictly alternating sequences of **P/D** and **T/S**. However, this doesn't ultimately change the underlying concepts ruling segmental sign subdivision, which continues to be in the mainstream of SL linguistic research.

2.2.1.5 Co-articulations

Co-articulation is the process of phonetic change where, after producing a phoneme, the organs involved in articulation adjust to produce the next one. In the case of SLs, this translates into the introduction of meaningless movements to the MHM sequence, as shown by Figure 2.7.

Sign			Co-articulation	Sign		
H_A	M_{AB}	H_B	M_{BC}	H_C	M_{CD}	H_D

FIGURE 2.7: The transition movement M_{BC}, between H_B and H_C, is a co-articulation.

Movements M_{AB} and M_{CD} shown above, are integral parts of their respective signs. On the contrary, M_{BC} exists just to serve as a stepping stone towards the articulation of H_C, coming from H_B.

Depending on the previous and next segments in the sequence, previously significant Movements can be reduced to co-articulations and vice-versa. Figure 2.8 shows a representation of this phenomenon.

Sign			Co-articulation	Sign		
H_A	M_{AC}	H_C	M_{CD}	H_D	M_{DB}	H_B

FIGURE 2.8: The transition movement M_{CD}, before part of a sign, is now a co-articulation.

In Figure 2.7, the sequence $H_C \rightarrow M_{CD} \rightarrow H_D$ forms a sign, which renders M_{CD} significant. However, the same sequence doesn't form a sign in Figure 2.8, turning M_{CD} into a meaningless segment. This happens despite the fact that, in both cases, M_{CD} appeared in the same order with respect to H_C and H_D. This goes on to show that the classification of Movements as co-articulations is not possible at the phonology, but rather relies on rules existing at the morphology.

2.2.2 Morphology and Lexicon

Morphology is the sub-discipline of linguistics that studies how the smallest meaning units, *morphemes*, combine to form *lexemes*, which are none other than words. It also studies how lexemes themselves are modified (*inflected*) by morphemes, combinatorially building semantic variations: for example, the lexeme WALK can be inflected into the forms *walks*, *walked* or *walking* [Booij 2005, pp. 8–9]. Together, lexemes form *lexicons*, which are essentially indexes of conventionalized words in a language.

In SL, morphology deals with the internal structure of signs in two ways: it studies both the multimodal nature of SL morphemes — arising from the use of different articulation channels —, and the use of signs as grammatical *affixes* [Aronoff 2005]. More intuitively, SL morphology studies both:

— how the alteration of direction, rhythm, or shape of a base sign, modifies its grammatical features;

— how sequentially concatenating signs (as prefixes or suffixes) to a base sign, affects its meaning.

Essentially, this duality in SL morphology yields an informal classification of signs, which divides them into easily identifiable groups for their study:

Lexical signs Also called *dictionary signs*, as these are the ones more likely to be found in dictionaries. Lexical signs have well-known meanings, as they make reference to specific concepts, objects and ideas. Roughly, these are the most closely related to vocal lexemes.

Grammatical signs These are the ones used as affixes, to modify properties of lexical signs. Broadly, they "insert" meaning to surrounding lexemes during discourse [Contini-Morava 2000, pp. xii – xiv]. For instance, they can be used to mark aspects such as pronouns, number, to differentiate arguments, etc. These kinds of signs are difficult to define out of context, in part due to their grammatical function.

Finger-spelled signs Refers to sequences of hand configurations and movements used to represent single alphabet letters. These signs let the signer spell out mostly nouns, with particular emphasis to loan words [Padden 2003, pp. 18–22].

Alternate classifications may exist in the literature, depending on the studied language; regardless, the basic rules of inflection remain consistent across different SLs, thus prompting further inquiries on the roles played by factors such as *space*, on different morphological case studies.

2.2.2.1 Space

Space plays an important role in SLs. It is through spatial relations that signers are capable of expressing complex reasoning [Emmorey 2003]. Accordingly, space has been observed to fulfill diverse linguistic roles, notably deictic, by providing *mappings* towards discourse entities and other *syntactical* functions [Padden 2010a]. Signers will commonly *map* spatial locations to represent persons, places or objects, which activates the referred space regions. This phenomenon is called *spatialization*, and serves to keep track of discourse referents; once defined, space locations remain active as long as the signers keep acknowledging their existence. This avoids the need of redefining a referent multiple times over the same context. For example, Figure 2.9 shows a signer performing the lexical sign $\texttt{FRIEND}_{\texttt{LSF}}$ and then pointing to the side, indicating that the space to his right will represent a friend.

$\texttt{FRIEND}_{\texttt{LSF}}$

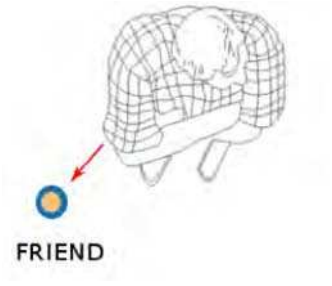
Spatialization to the right

Upper-view of active locations

FIGURE 2.9: Spatialization of $\texttt{FRIEND}_{\texttt{LSF}}$ to the right of the signer.

Active space locations let signs *agree* with them, so as to add new information about the referents. For instance, signs like $\texttt{CALL}_{\texttt{LSF}}$, $\texttt{HELP}_{\texttt{LSF}}$, $\texttt{TELL}_{\texttt{LSF}}$ or $\texttt{PAY}_{\texttt{LSF}}$, agree with space by signaling or moving towards active locations [Hong 2006], in order to indicate their *grammatical case* (*e.g.* who receives the action, who executes it, etc). For example, if the signer performs $\texttt{CALL}_{\texttt{LSF}}$ by moving his hand towards his right, as shown on Figure 2.10, the utterance would be translated as "call a friend". Signing $\texttt{CALL}_{\texttt{LSF}}$ from the right side and moving towards himself, it would indicate "a friend calls me".

$\texttt{CALL}_{\texttt{LSF}}$



Upper-view of agreement to a location

FIGURE 2.10: Hand agrees with space location to produce "Call a friend".

Similarly, new characteristics of the referred entity can be introduced by making reference to the active spatial location. Following the same example, if the signer performs $\texttt{UGLY}_{\texttt{LSF}}$ and points towards his right side, the utterance would translate as "friend is ugly".

In addition to referent localization and agreement, space in the morphology fulfills *topographic* functions, by permitting to specify how objects are physically located with

respect to one another. For instance, in several SLs it is possible to express "a cat is over the fence" by assigning a location for *fence* and then signing *cat* just above [Engberg-Pedersen 1993, Meir 1998, MacLaughlin 2000].

2.2.3 Sign Language Semantics

Semantics deals with *linguistic signs* [Saussure 1916]; in particular, with how meaning is established and conveyed by language. In SL, this process is deeply linked with form and space since, contrary to vocal interlocutors, signers can use their bodies to provide rich visual representations of existing objects, situations and people, which in turn can produce complex discourse.

2.2.3.1 Iconicity

The term *iconicity*, refers to how the form of a sign, an *icon*, is directly linked to the meaning of the sign itself. Contrary to vocal languages, SLs rely heavily on iconicity, often as a central part of communication [Liddell 2003]; the same behavior can be seen in vocal phenomena such as *onomatopoeia*, but it is less common than what it has been observed in SL.

Iconic gestures in SL are seen in different discourse levels. The most basic one, directly relates sign postures to real life objects by mimicking their characteristics. An example can be seen on Figure 2.11.

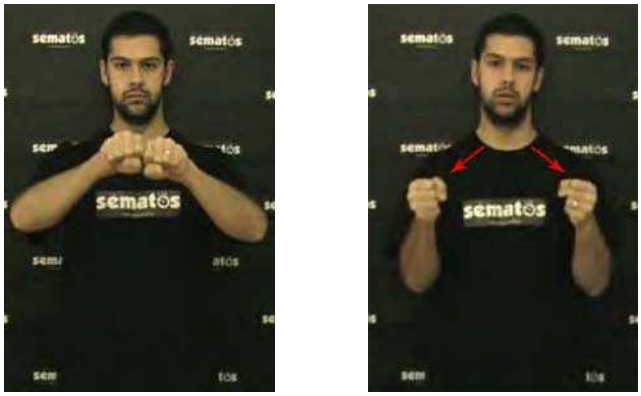
TREE$_{\text{LSF}}$

FIGURE 2.11: TREE$_{\text{LSF}}$ is an example of an iconic sign.

The sign TREE$_{\text{LSF}}$, depicted above, is executed by holding the dominant hand parallel to the body. At the same time, the hand retains a completely open hand configuration.

The form of the sign is clearly influenced by its meaning: the arm represents the trunk of a tree, and the open hand its foliage.

Iconicity is present in a large number of SL lexical signs, but not always with direct relation to physical objects; sometimes, the iconicity of a sign is related to a metaphor rather than to the form of a real, physical, object. An example is given by sign $\texttt{PAUSE}_{\texttt{LSF}}$, presented in Figure 2.12.



$\texttt{PAUSE}_{\texttt{LSF}}$

FIGURE 2.12: $\texttt{PAUSE}_{\texttt{LSF}}$ uses iconicity in a metaphoric way, by representing a pause as a physical "break".

The sign $\texttt{PAUSE}_{\texttt{LSF}}$ is performed by holding both hands together as if they were grasping an object, and then separating them abruptly as if breaking said object. The form of the sign relates then to the meaning of "break", which is metaphorically linked to the concept of "pause", but doesn't really involve the physical act of breaking something.

The iconic use of space is also a notorious characteristic of SLs [Padden 2010a]. For instance, to describe that a recipient is big, the signer can execute the sign $\texttt{RECIPIENT}_{\texttt{LSF}}$ with very exaggerated movements. Similarly, to express that an object is small, the signer will perform the sign with very short movements, delimiting the surface of the proposed object. In the same way, signers can outline the size of buildings, surfaces or spatial points of reference, which most of the time will remain active in space, in the position they were originally located. In this way, signers can describe the distribution of an auditorium, of a neighborhood, or any other complex scene, where they can either place themselves or other referents in order to establish complex relationships [Engberg-Pedersen 1993]. In a way, this transforms spatial regions into objects, which can be modified directly during discourse (similar to what happens in discourse deixis of vocal languages [Lyons 1977]). For example, the signer can "take" the recipient he originally placed in space (by mimicking the action of taking the object), and locate it elsewhere (by mimicking the release of the object in a different place).

Finally, the maximal expression of SL iconicity is shown by the *role shifting* phenomenon [Cuxac 2000], where signers "act the part" of a discourse referent by taking their place and assuming their actions as their own [Lillo-Martin 1995]. In this way, the signer can directly show what a referent did or said from the referent's perspective.

During a role shift, the signer completely steps out of his own persona in order to mimic the behavior of the referent. In this situation the entire body expression becomes iconic, as it is completely influenced by the characteristics of the role taken. Figure 2.13 shows an example of role shifting, where the signer narrates a story about a fire and takes the role of a firefighter breaking a door.

FIGURE 2.13: The signer shifts roles to narrate his story from the point of view of a firefighter.

Putting everything together, it is easy to see that meaning in SL signs is deeply linked to multimodality at every level of study: it depends heavily on how signers interpret different forms — created by several articulators working simultaneously — over many layers of abstraction. As such, it becomes important to determine what individual functions each articulator may have, how they are modified to convey the images needed to build these forms and how they manage to produce unambiguous messages, as explained in the next section.

2.2.4 Parametric Sign Language Articulation

The information produced by utterances, can be modeled by way of discrete *articulation parameters*. Depending on their physical characteristics, articulators provide a set of different parameters in order to convey information. The hands will invariably produce a great amount of the overall message, since they are very flexible and can change their form easily. However, this is not to say that the information provided by Non-Manual Features (NMFs) is any less important; research shows that native signers identify signs faster if they have NMFs in them than if they don't [Wilbur 2009].

There are five classical parameters used in SL description [10], based on the seminal work by [Stokoe 1960]. They are enumerated here as follows:

1. **Hand configuration**.- Refers to the shape of the hand while signing. Due to the high flexibility of hands, the quantity of available configurations can be very high. However, observations on various SLs have shown that only a relatively small subset of the possible configurations is commonly used implying that, at least in some cases, the quantity is not as big as it could be [Mann 2010]. Figure 2.14 shows some common hand configurations used in ASL.

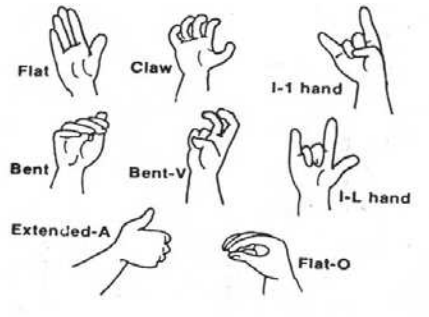


FIGURE 2.14: Hand configurations used in ASL. [11]

2. **Hand orientation**.- This parameter indicates the direction in which the hand is "pointing". This can be characterized more formally as the direction of a vector perpendicular to the plane formed by the palm is pointing, as shown in Figure 2.15. Orientation can be defined in terms of other hand sections, however the palm and the back of the hand are the most common [Hong 2006].

FIGURE 2.15: Hand orientation vector with respect to the palm.

3. **Articulation place**.- It refers to the body or spatial region where the sign is being executed. As previously stated, not all articulator places are located on the signer's body.

10. The list of parameters may be slightly different depending on the studied SL.
11. Image obtained from [Venkatagiri 2014].

4. **Hand movement**.- Indicates the characteristics of hand motion. These can be as simple as straight movements like the ones shown in Figure 2.16, or as complex as irregular trajectories describing the physical form of an object. This parameter also takes in account salient movement characteristics such as speed or acceleration.

   More complex phenomena such as *repetition* can also be described as part of this parameter; that is, whether a particular motion is executed just once, or several times as part of a single movement.

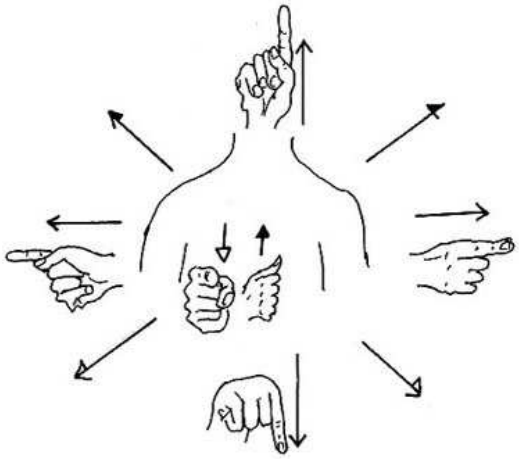


FIGURE 2.16: Subset of the possible movements a hand can execute during signing. [12]

5. **NMFs**.- Refers to facial expression, head, shoulders and overall body motion, occurring simultaneously with hand articulation.

As the existence of single-handed signs implies, not all articulators participate on the production of all signs; in the majority of cases, only a subset of the possible parameters is necessary to distinguish each. For this reason, some time-units may lack of movement, NMFs or even hand interaction, at all. This doesn't affect the underlying principle of sign formation but may, however, complicate low-level observations, due to the sheer amount of possible parameter combinations [Liddell 1989]. This phenomenon will be apparent in the following chapters, where it will be explained how this combinatoric nature affects distinguishability of individual postures, which in turn can become an issue both for both representation and automatic recognition.

12. Image obtained from [Venkatagiri 2014].

2.3 Chapter conclusion

This chapter has presented the main characteristics of SL linguistics, with special emphasis on how SLs are most commonly described at the phonetic level. SLs are the visual-gestural languages that members of the Deaf community use to communicate. They differ from vocal languages in their production, as they are articulated in space by way of corporal movements. The complexity of their articulation parameters and phonological features, make them interesting subjects of study both in linguistics and computer science. However, their very particular characteristics make them difficult to transcribe by way of written notations, which can pose problems to automatic analysis tools.

Similar to vocal languages, SLs are usually studied with the help of corpora: digital compilations of language samples documenting their use, usually annotated with external data for their processing. However, because of the difficulty to represent them otherwise, SL corpora are forced to remain as video or motion capture collections, which have to be treated with specialized computer-based tools for their processing.

Interest in the use of software tools for the study of SL has sparked new research ventures, specially in areas such like SL recognition and SL synthesis. However, the enormous quantities of possible posture combinations — and the physical characteristics of articulators — pose a number of different problems that need to be solved before new automatic solutions can be developed. This is specially true for phonetic representation, which ultimately serves as the basis of language construction. For this reason, this thesis is exclusively centered around representation issues on the phonetic level, and it proposes a flexible phonetic description framework that can be used within computed-aided SL research tasks.

With this in mind, the next chapter presents a review of current research in computer science, aimed towards the study of SLs. Emphasis is put on how new projects tend to work on problems related to the recognition and, specially, representation of segmental features and signing parameters, with perspectives on building high level solutions from these smaller projects in the future.

Chapter 3

State of the Art

As interest in the study of Sign Language (SL) has increased, the development of new computer-based techniques for their analysis has become unavoidable. However, contrary to what happens in vocal language research, SL projects have to deal with corpus representation challenges from the very beginning. For example, discussions about adequate corpora formats [Crasborn 2008c, Braffort 2010, Hanke 2010] and multimodal annotation standards [Martell 2002, Dreuw 2008a, Schembri 2010] are recurrent issues in the area, showing that the domain is still finding its footing. Thus, a lot of effort has been put into studying SL representation across different domains, since it has become apparent that it may be a crucial part of corpus-based SL research.

This chapter presents the existing state of the art for computer-based SL processing. The first half introduces the current research efforts related to the domain, seen from the point of view of SL research; the second half, presents some of the computer science tools on which the contributions of this dissertation are based.

With this in mind, the next sections introduce some of the most common SL transcription systems used in linguistic research. Moreover, Sections 3.2 and 3.3 give a brief introduction to SL synthesis and SL recognition, respectively, showing how representation needs may change between different research efforts. Finally, Section 3.4 gives a brief introduction to *formal methods*, presenting how systems engineering deals with representation issues in the development of industrial applications, always following well-established patterns despite the fact that application domains may differ from one another.

3.1 Sign Language Representation

SL representation research, consist in the development of formal notation systems for the transcription of SL utterances. These notations can range from written SL systems, to phonetic transcription languages. Currently, several representation notations exists for different objectives, but neither of them can be considered as standard in their respective domains of application. Nonetheless — and regardless of their intended use — most of the existing SL representations share various common characteristics, reflecting the importance of capturing the underlying articulation parameters (see Chapter 2, section 2.2, p. 24) as accurately as possible.

With respect to SL notations for linguistic research, interest has hovered around the development of good transcription formalisms, in an effort to reduce reliance on video materials. Also, this road have been explored as a way to potentially increase the application of Natural Language Processing (NLP) algorithms to SLs corpora [Miller 2001, van der Hulst 2010]. However, the existence of contradicting views in favor [Hoiting 2003] and against [Johnston 2010] transcription, further shows the relative instability of the area, blocking the adoption of transcription as a widespread method of analysis.

Despite opposition to thorough phonetic transcription, it is often necessary in research to describe detailed SL sub-lexical information for recognition and synthesis. This proves to be specially important for the latter (see Section 3.2 below), as the existing SL synthesizers tend to use some form of notation to encode their inputs. This has led to the development of very specialized description tools, which have been integrated to corpus annotation software like [Hanke 2002], for example. As a consequence, sign description systems like the Stokoe Notation or Hamburg Notation System (HamNoSys) have gained popularity among researchers, as they have served as the basis of various SL projects on different domains.

In the following sections, some of the existing notations used to represent SL in research tasks are described. In addition, a small presentation of SignWriting — a written SL system for common day-to-day use — is given, in an effort to contrast how SL representation may change, depending of its intended use.

3.1.1 Stokoe Notation

Before the year 1960, it was usually assumed by linguists that SL signs were indivisible, incapable of being represented by way of separable units like sounds in vocal language words. However, with the publication of the paper by [Stokoe 1960], a new paradigm in SL studies was born, where signs could be effectively "cut" into subunits for their

description. This enabled the author to develop his own transcription system, the *Stokoe Notation*, aiming to capture signing in a linear structure akin to written vocal language.

The author designed his notation to be employed as a tool for the linguistic study of SLs. It provides a set of typewritten symbols, used to represent articulation place, hand configuration and movement. Figure 3.1 shows a subset of these symbols.

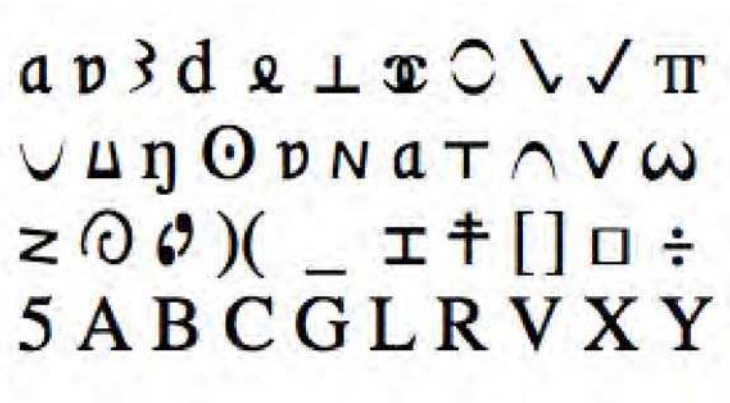


FIGURE 3.1: Subset of Stokoe Notation symbols.[1]

Broadly, sign transcriptions are written linearly and with a fixed character order. An example is shown in Figure 3.2, where the Stokoe Notation was used to write the sign $\texttt{SNAKE}_{\texttt{ASL}}$.

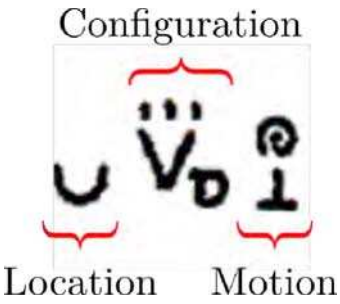


FIGURE 3.2: $\texttt{SNAKE}_{\texttt{ASL}}$ in Stokoe Notation.

In the figure, the first character denotes location; the U-like symbol indicates that articulation takes place over the lower part of the face. The second character describes the hand configuration: a finger-spelled "V". The three points above the "V" means that the fingers are bent, and the inverted "a" denotes that the back of the hand is facing upwards. Finally, the last group of characters denotes an outwards circular motion of the hand.

Because of the technological limitations at the time of its creation, the Stokoe Notation is limited to 55 Latin characters. For this reason, some symbols change meaning depending

1. Image obtained from [ASLFont 2013].

on their place in the chain. The disadvantage of using such a limited set of characters is that the resulting descriptions become highly encoded and barely iconic with respect to the actual signs, which can impact readability. In addition, non-standard characters are often needed to describe articulation parameters of SLs other than American Sign Language (ASL), which can further complicate the comprehensibility of the notation.

Even with its disadvantages, the Stokoe Notation proved to be a practical way to encode SL, a feat showcased by the author with the creation of a detailed ASL dictionary [Stokoe 1976]. Later projects have tried to conserve this practicality by following the same representation strategy, but taking advantage of new technologies to create specialized symbols rather than just re-purposing Latin characters. In this manner, new notations like HamNoSys improve the readability of their sign descriptions by using more iconic symbol sets, closer to the actual visual form of the represented sign.

3.1.2 HamnoSys

HamNoSys is a phonetic SL transcription system. Like the Stokoe Notation before, it was originally created for linguistic research; it was not intended to be used as a SL writing system for daily life, but rather as a tool to represent research resources [Hanke 2004]. To this end, HamNoSys defines a greater amount of notation symbols; it holds around 200 characters, intended to represent a wide range of parametric postures in several SL.

The symbols defined by HamNoSys are more iconic than those by Stokoe. For instance, the representation of hand configurations and places of articulation retain a visual relation with the actual morphological characteristics of the human body. Figure 3.3 shows some of the characters used by HamNoSys to represent common places of articulation and hand configurations.

Like the Stokoe Notation, HamNoSys descriptions are written as chains of positional characters. One of the differences between the two, however, is that in HamNoSys symbols are highly specialized, so they'll usually retain their meaning regardless of their position on the chain. In this case, positioning serves rather to enable the automatic processing of the transcribed data; computer systems are able to parse sign descriptions without ambiguity, as the positional structure is known. This advantages have proven useful for the creation of HamNoSys-based annotation software [Hanke 2001] and for the development of avatar-based synthesis technology [Elliott 2004].

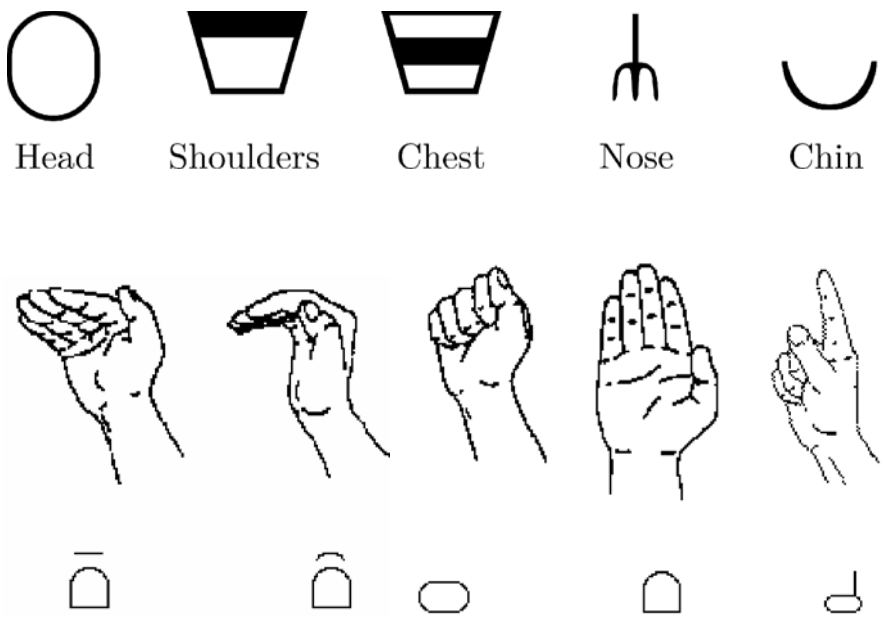


FIGURE 3.3: Iconic places of articulation and hand configurations in HamNoSys.[2]

Similarly, HamNoSys centers around hand configurations, places of articulation and movements; however, it is more expressive than Stokoe notation, as it can express very nuanced constructions by way of modification symbols. Figure 3.4 shows an example of a German Sign Language (DSG) sign, $\texttt{HAMBURG}_{\texttt{DSG}}$, represented in HamNoSys.

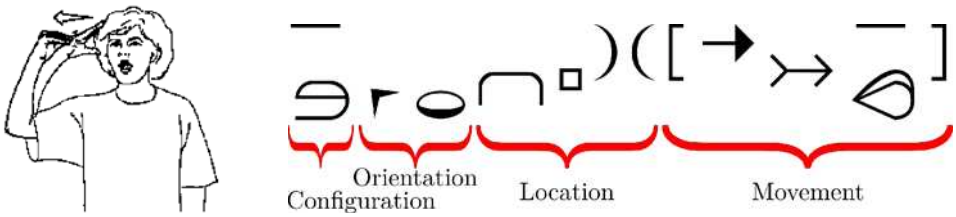


FIGURE 3.4: $\texttt{HAMBURG}_{\texttt{DSG}}$ in HamNoSys.[3]

As with hand configurations and locations, HamNoSys's movements are very iconic and can have many variations. Figure 3.5 show an example of some of the symbols used to depict movement in HamNoSys.

Even though HamNoSys tends to be more iconic than the Stokoe Notation, it still depicts signs with a high level of abstraction. Furthermore, as it is intended to be used as a tool for linguistic research, signs are usually described with great detail. This results in long and complex chains of symbols, which in some cases can become difficult to read and write without help of an expert.

2. Images obtained from [Prillwitz 1989].
3. Images obtained from [Prillwitz 1989].

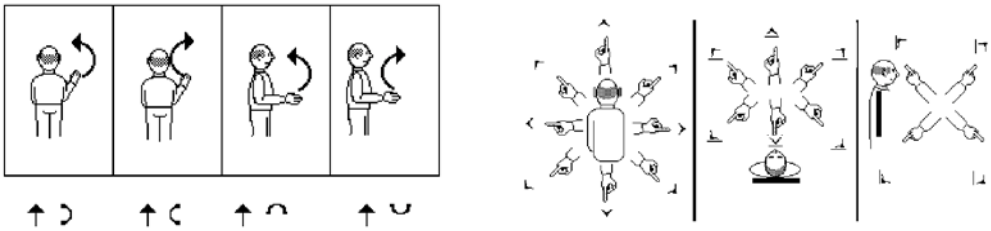


FIGURE 3.5: Movement and direction symbols in HamNoSys. [4]

A slightly different approach to these kind of representation systems is depicted by the SignWriting notation, which was developed to be used as a writing system for SLs rather than for linguistic research. For this reason, SignWriting avoids describing symbols in a sequential order in favor of highly iconic structures, in an effort to visually represent the physical formation of signs.

3.1.3 SignWriting

SignWriting is a SL representation system inspired on a notation originally developed to capture Ballet movements [Sutton 2000]. The main principle behind it, is for descriptions to be both precise and easy to read. Contrary to HamNoSys and the Stokoe Notation, SignWriting is effectively used by the ASL Deaf community as a written communication system and not only as a tool to represent SL in research.

Similarly to the Stokoe Notation and HamNoSys, SignWriting describes phonetic parameters by way of a set of symbols, defined to represent hand configurations, places of articulation and movements. However, SignWriting doesn't describe signs as chains of symbols but rather as very iconic *glyphs*, representing postures as they would be "seen" from the point of view of the signer. An example of a glyph can be seen on Figure 3.6.

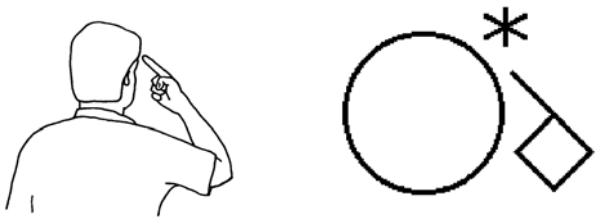


FIGURE 3.6: A SignWriting glyph describing a posture where the signer's dominant hand touches ($*$) the head ($\circ$), while holding a `POINTING` configuration. [5]

4. Images obtained from [Prillwitz 1989].
5. Images obtained from [Sutton 2009].

In the figure, the circle represents the back of the head, giving an spatial frame of reference for the depicted signer. The white square means that the back of the hand is oriented outwards, while the asterisk symbol (∗) represents a simple touch to the head. Figure 3.7 shows other possible contact symbols.

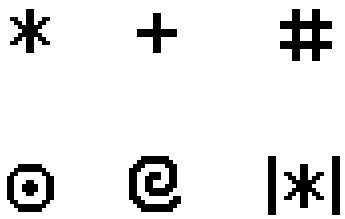


FIGURE 3.7: SignWriting contact symbols to represent "touch", "grasp", "strike", "brush", "rub" and "enter".

SignWriting glyphs rely on the morphology of the human body to establish spatial points of reference by drawing simple figures. The head, for example, is represented by a big circle, usually featured prominently. From there, places of articulation such as the shoulders, face or others are defined with respect to the head. This, in combination with special modification symbols, serve to represent complex parameters such as Non-Manual Feature (NMF) or body movement in a very intuitive way. An example of the expressive capabilities of this method can be seen in Figure 3.8.

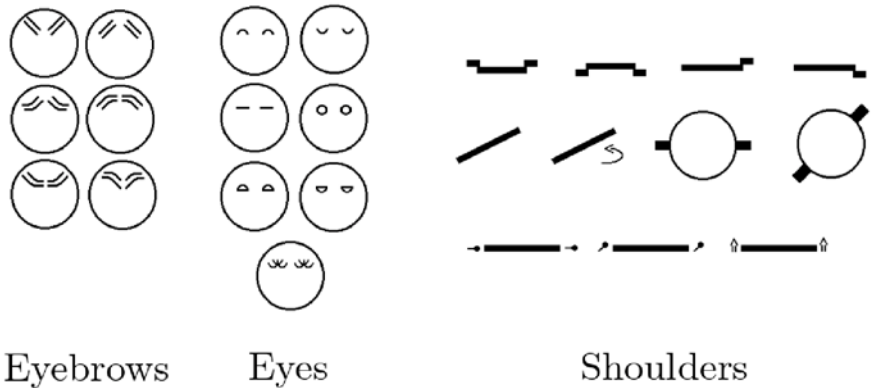


FIGURE 3.8: SignWriting places of articulation and their possible modificators.[6]

As with locations, hand configurations in SignWriting are very iconic, even capturing the position of the fingers in the configuration glyphs. Figure 3.9 shows some of the symbols used to describe hand configurations.

6. Images obtained from [Sutton 2009].

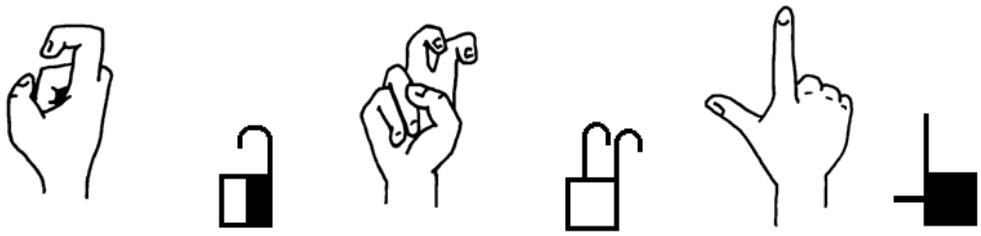


FIGURE 3.9: SignWriting hand configurations. [7]

On the other hand, movement representation remains fairly similar to HamNoSys, using distinct types of arrows to differentiate between types of movement. Figure 3.10 show the difference between vertical and horizontal movements.

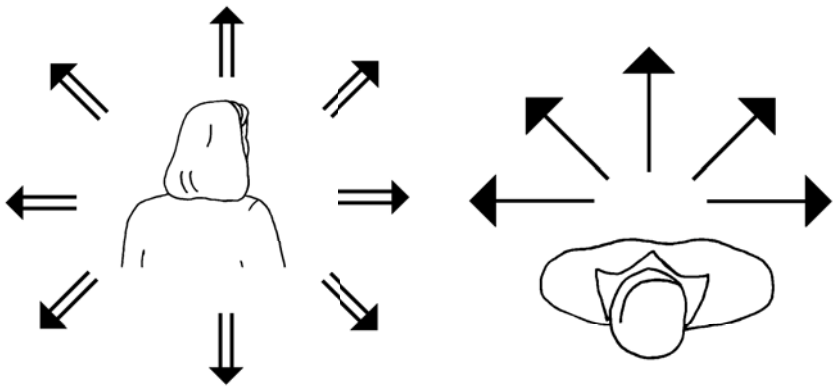


FIGURE 3.10: SignWriting vertical and horizontal movements. [8]

Even though they are represented similarly, SignWriting is able to take advantage of glyph structure to project movements directly unto their place of articulation on a single image. For example, the Figure 3.11 shows the representation of sign $\texttt{ADDRESS}_{\texttt{LSF}}$. The glyph in the figure is able to show the articulation place, hand configuration and a complex movement as a unit. In this case, both hands hold configuration `THUMB` and execute a descending movement in parallel over the shoulders (represented by the black bar under the head). The color modificators in the configuration (black and white), indicate the orientation of the hands (palms perpendicular to the body, facing inwards). The contact symbols indicate that hands "brush" the signer's body while executing the movement, and the double arrows indicate that the action is repeated twice.

7. Images obtained from [Sutton 2009].
8. Images obtained from [Sutton 2009].

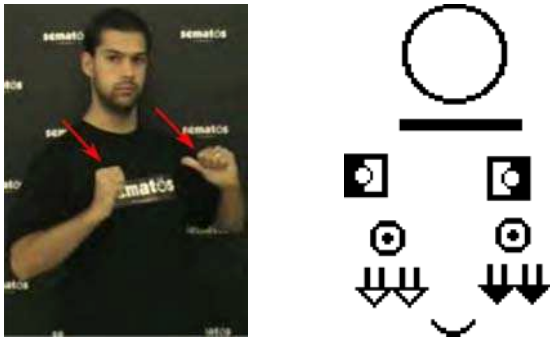


FIGURE 3.11: Sign $\texttt{ADDRESS}_{\texttt{LSF}}$ represented in SignWriting. [9]

SignWriting has proven to be easier to read than other representation systems, which has positively influenced on its adoption by Deaf communities [Flood 2002]. From the point of view of technology, SignWriting glyph editors [Borgia 2015] and SignWriting-based pedagogic software exist commercially since the late 90's. However, despite its advantages, this notation is not optimal to be exploited in research, in part because of it's unconventional, not trivial to parse, structure [Koller 2013]. For this reason, research projects relying on SL representation tend to disregard very iconic notations like SignWriting in favor of more technical ones like HamNoSys, something which rings specially true in areas like SL synthesis, where very detailed sign descriptions are the norm.

3.2 Sign Language Synthesis

As previously introduced, SL synthesis deals with the artificial generation of language utterances by virtual signers (*avatars*) [Kipp 2011]. One of the main interests in the area is to build avatars capable of mimicking the nuanced parametric changes made by signers during discourse [Huenerfauth 2007]. This is because detailed parameters can be crucial for the unambiguous identification of phonemes, due to their combinatoric nature (see Chapter 2, Section 2.2.1, p. 25).

In an effort to confer a more expressive quality to avatars' gestures, research in synthesis has tended to move towards data-driven approaches [Rezzoug 2006]. For instance, the work presented by [Cox 2002] proposed the development of a SL translation system, aimed to improve interaction between Post Office clerks and Deaf users. Broadly, the authors describe the construction of an automatic translator, used to transform a clerk's speech into synthetic British Sign Language (BSL) (produced by an avatar). In order to enhance intelligibility, avatar's productions are animated as accurately as possible by way

9. Signer image obtained from [Sématos 2013] and SignWriting description obtained from [Sutton 2010].

of detailed human movement data. To this end, the researchers assembled a custom-made motion capture rig from different body bound sensors, including a head-mounted camera to capture NMFs. The rig served to build a small corpus, compiling data from multiple human signers, which was then used to generate realistic signing from the avatar. The quality of the animation was tested by synthesizing a set of 115 BSL phrases, which were given to a group of native signers to decide whether they were comprehensible or not. In their experiments, the researchers found that the average comprehensibility was situated between 42% and 70%. However, most of the negatives were influenced by linguistic variation rather than technical issues: in some cases, signers didn't recognize phrases because they were signed differently to what they were used to — not because the animation was flawed — further underlining the importance of linguistic research in the development of synthesis technology.

On the same vein, works like [Huenerfauth 2007, Lu 2010] present similar corpus capture and evaluation methods, oriented towards the development a better suited corpora for both synthesis and machine learning tasks. However, similar observations to those of [Cox 2002], regarding comprehensibility and utility of the obtained data, are expressed by the authors, pointing towards the lack of sufficient linguistic knowledge as one of the main reasons behind poor results in these kind of systems.

In this sense, some synthesis research has centered around issues other than improving data capture. This efforts go from the development of tools for the edition of existing corpora [Lefebvre-Albaret 2013], to the development of new representation models [Losson 1998] intended to enrich corpus information. For example, [Karpouzis 2007] presents the creation of a synthesis system's architecture for Greek Sign Language (GSL), oriented towards education. For its development, a number of GSL resources were collected through various projects. This lead to the creation of a grammatical model of GSL, intended to be used in the production of phrases. Additionally, the project includes a lexical representation database encoding individual signs with HamNoSys. Despite their efforts, researchers point out that their approach is limited by both lack of enough standard linguistic resources for GSL, difficulties with the technicality of the chosen representation system, and problems linked to the production of co-articulations, which lead to unintelligibility.

Finally, works like [Awad 2009, Duarte 2010], propose the analysis of corpus annotations to improve the quality of virtual signer animations. In particular, the authors define a custom annotation schema for their project, taking in account elements such as grammatical markers from facial expressions; hand-movement descriptions; deictic uses of space and, notably, the occurrence of co-articulations.

The adoption of technology standards to encode their proposed schema — like the Extensible Markup Language (XML) [Bray 1997] — enables great flexibility on studying which changes depicted in the video are relevant and when: each human-identified sign in corpora can be decomposed into the different aspects defined by the schema, making possible not only the generation of motion profiles from data, but also the development of rules to encode how signers synchronize their articulations depending on what's being signed. In this manner, synthesis can be greatly improved by linguistic knowledge of sign morphology, by showing how they effectively form depending on the context of communication. However, the disadvantage of this kind of studies, is that they are heavily dependent on the existence of quality linguistic annotations; as previously said, this is not necessarily a given for every corpus (see Chapter 1, Subsection 1.1.1, p. 3). In addition, this also points out the limitations that notations such like HamNoSys have in synthesis; they usually have to be modified into more technical representations capable of this kind of schematic decomposition, as it is presented in the next section.

3.2.1 Sign Language Representation in Synthesis

SL representation notations in synthesis are usually built depending on their purpose inside the system. In some instances, descriptions will serve as thoroughly geometrical recipes, detailing how a sign has to be animated [Elliott 2004]. In others, representation formalisms can serve as higher level models [Filhol 2012b], introducing linguistic information to the process [Lejeune 2002]. In most cases, there will necessarily be an overlap between the two, due to the importance of linguistic data for synthesis.

An example of the combination between linguistic and geometrical descriptions is given by [Karpouzis 2007]. In their system, lexical sign representations are written in HamNoSys, while avatar movement descriptions are defined in the Scripting Technology for Embodied Persona (STEP) language [Huang 2004]. A STEP description is shown in Figure 3.12, where the recipe of how to build a hand configuration is presented.

In general, their system parses HamNoSys descriptions into STEP instructions: each HamNoSys symbol corresponds to a set of STEP scripts, indicating the avatar how to render sequences of postures. In this manner, linguistic resources created with HamNoSys can be synthesized directly without having to give geometrical descriptions to the system; as there are already STEP-conversion rules in place to do it automatically. Despite this advantage, the authors mention that their method could prove troublesome to extend beyond manual parameters, as articulation from NMF tends to be more nuanced and, thus, less adapted for a straightforward conversion.

```
par([
    turn(humanoid,r_thumb1,rotation(1.9,1,1.4,0.6),very_fast),
    turn(humanoid,r_thumb2,rotation(1,0.4,2.2,0.8),very_fast),
    turn(humanoid,r_thumb3,rotation(1.4,0,0.2,0.4),very_fast),
    turn(humanoid,r_index1, rotation(0,0,0,0),very_fast),
    turn(humanoid,r_index2,rotation(0,0,0,0),very_fast),
    turn(humanoid,r_index3,rotation(0,0,0,0),very_fast),
    turn(humanoid,r_middle1,rotation(0,0,1,1.5999),very_fast),
    turn(humanoid,r_middle2,rotation(0,0,1,1.5999),very_fast),
    turn(humanoid,r_middle3,rotation(0,0,1,1.5999),very_fast),
    turn(humanoid,r_ring1,rotation(0,0,1,1.7999),very_fast),
    turn(humanoid,r_ring2,rotation(0,0,1,1.5999),very_fast),
    turn(humanoid,r_ring3,rotation(0,0,1,0.6000),very_fast),
    turn(humanoid,r_pinky1,rotation(0,0,1,1.9998),very_fast),
    turn(humanoid,r_pinky2,rotation(0,0,1,1.5999),very_fast),
    turn(humanoid,r_pinky3,rotation(0,0,1,0.7998),very_fast)
    ])
```

FIGURE 3.12: STEP code for a given hand configuration.

3.2.1.1 Signing Gesture Markup Language

A similar approach is followed by the Signing Gesture Markup Language (SiGML) [Elliott 2004], a representation formalism based on HamNoSys. The main difference between them, is that SiGML is designed to work as a XML standard on its own. An example of a SiGML description is shown on Figure 3.13.

As with previous representation efforts, SiGML is based on segmental phonology models, such as the one presented by [Liddell 1989]. However, unlike other formalisms, with SiGML the authors strive to combine sign parameters into morphological structures: each parameter description can be either interpreted all by itself, as a phoneme of sorts, or it can be grouped with others to be identified together as a lexeme. In practice, this means that avatars may execute either an isolated parameter (single description file) or complete signs (group of description files), by controlling the groupings of individual parameters. Also, XML has the technical advantage of having a well-defined, nested, structure; this simplifies the addition of new elements to the schema if needed, which can further adapt the language to contain more information than HamNoSys.

This inherent adaptability has resulted into the creation of more flexible representations: it has enabled the combination of multi-dimensional HamNoSys strings, representing higher level linguistic functions [Elliott 2007]. In turn, this has opened the door to the exploration of synthesis at higher levels where avatars may, for example, receive instructions in term of linguistic functions (*e.g.* "make reference to") rather than geometric instructions. Nonetheless, some expressiveness problems may hamper efforts in this direction, due

```
<?xml version="1.0" encoding="utf-8"?>
<sigml>

<hns_sign gloss="mug">
<hamnosys_nonmanual>
<hnm_mouthpicture picture="mVg"/>
</hamnosys_nonmanual>
<hamnosys_manual>
<hamfist/> <hamthumbacrossmod/>
<hamextfingerol/> <hampalml/>
<hamshoulders/>
<hamparbegin/> <hammoveu/> <hamarcu/>
<hamreplace/> <hamextfingerul/> <hampalmdl/>
<hamparend/>
</hamnosys_manual>
</hns_sign>

<hns_sign gloss="take">
<hamnosys_nonmanual>
<hnm_mouthpicture picture="te_Ik"/>
</hamnosys_nonmanual>
<hamnosys_manual>
<hamceeall/> <hamextfingerol/> <hampalml/>
<hamlrbeside/> <hamshoulders/> <hamarmextended/>
<hamreplace/> <hamextfingerl/> <hampalml/>
<hamchest/> <hamclose/>
</hamnosys_manual>
</hns_sign>

<hns_sign gloss="i">
<hamnosys_nonmanual>
<hnm_mouthpicture picture="a_I"/>
</hamnosys_nonmanual>
<hamnosys_manual>
<hamfinger2/> <hamthumbacrossmod/>
<hamextfingeril/> <hampalmr/>
<hamchest/> <hamtouch/>
</hamnosys_manual>
</hns_sign>

</sigml>
```

FIGURE 3.13: SiGML description of the sign $\texttt{SCOTLAND}_{\texttt{BSL}}$ in BSL.

to the fact that HamNoSys was originally created for phonetic transcription; this is, by design it lacks the necessary structures for syntactic representation, and thus it may require multiple extensions to be able to support it.

3.2.1.2 Zebedee

A different approach to SL transcription for synthesis comes in the form of Zebedee [Filhol 2009]. This formalism, is based on the phonetic model presented by [Johnson 2011] (see Chapter 2, Subsection 2.2.1.4, p. 29).

For Zebedee, segment order comes above purely parametric descriptions, making special emphasis on time constraints such as *duration*. In addition, sign descriptions in Zebedee are *dynamic*: parameters don't remain fixed over time, but are *dependent* of internal changes that may change their values over time. By establishing dependency relations, experts describing signs may center around modeling sign structure, rather than having to thoroughly specify every possible parameter in the resulting synthesis; dependencies take care automatically, without further intervention. The same mechanism, enables a signing avatar deducing which movements perform, as dependencies make every one of them deducible from descriptions. Figure 3.14 shows an example of a Zebedee description.

As shown in the figure, Zebedee enables the inclusion of typed variables, which can provide specific information about independent articulators, places of articulation, and dependency relationships between them. In addition, Zebedee makes use of space by giving experts the possibility of mapping variables to spatial locations, which in turn permits the definition of inflections under the same formal language. In this manner, lexical signs given to the avatar may be directly modified during synthesis, incidentally simplifying the system's input. With this, and contrary to purely phonetic approaches, sign descriptions can remain very succinct, while still being able to generate very expressible output in practice.

3.2.1.3 Higher level representations

Most representation languages for synthesis have centered around phonetic transcriptions, however some work exists towards the representation of higher level structures. For instance, [Lebourque 1999] presents the development of a high-level specification, permitting the representation of spatial locations and the avatar's relation towards them. The proposed formalism is powerful enough to represent simple phrases, however the notation remains fairly low-level in terms of movement and synchronization.

```
SEQUENCE "ballon"

<language=LSF>
<numvidlimsi="1014_tout.mov">
<refdico="aucune">
<described_by="Flora, ">

DEP loc = @ABST + <FWD | medium>
DEP size = medium
Alias %p0 -> [loc] + <UP | [size]>

KEY_POSTURE(0){
KEEP:
For $h=s,w
#all4_contact($h)
Orient NRM!palm($h) along <@PA($h), [loc]>
End

HERE:
Place @PA(s) at %p0
Place @H_BACK(w) at @I_KNi(s,1)
}

TRANSITION(10){
Accel 1
Arc @PA(s) : <LAT | [size]>
Arc @PA(w) : - <LAT | [size]>
}

KEY_POSTURE(0){
HERE:
Place @PA(s) at SYM %p0 wrt [loc]
Place @PA(w) at @L_KN(s,1)
}

End "ballon"
```

FIGURE 3.14: Zebedee description for $\texttt{BALLOON}_{\texttt{LSF}}$.

Works like the one presented by [Lejeune 2002] go into more abstract representations, defining a set of linguistic operations and encoding contexts and verbs' meanings into formal structures. In this way, high-level linguistic phenomena such like iconicity could be represented easily in synthesis tasks. However, the disadvantage of working with such high-level representations is that it loses flexibility when confronted with existing synthesis engines, as they tend to need very verbose instructions to work with [Huang 2004].

Finally, improvements on the original Zebedee language — previously introduced — have resulted in the development of AZee [Filhol 2012a], a higher level representation language intended better accommodate multimodality in synthesis transcriptions. AZee permits the definition of *synchronization rules* inside descriptions, inducing an ordering of the articulatory information on different channels (see Chapter 2, Subsection 2.2.1.3, p. 28). This enables the system to synchronize multimodal information into single blocks containing more than one posture, therefore enabling representation of more complex linguistic phenomena such as *spatialization*, or the modeling of grammatical markers such like questions.

3.3 Sign Language Recognition

Due to the particularities of their linguistics (see Chapter 2 Section 2.2, p. 24), SL recognition tasks tend to be very complex to achieve; in order to recognize discourse, an automatic system needs to know how to make individual sense of both phonetic and morphologic SL characteristics, as well as more complex phenomena such as iconicity or the different uses of space [Bauer 2000]. For this reason, most of the existing research in the area has centered around the first two levels of linguistic study, in an effort to approach the problem in a bottom-up fashion [Ong 2005]. As such, researchers tend to subdivide their research efforts into hierarchical sub-disciplines, reflecting the different levels used in linguistic research:

Sub-lexical recognition Work in this level consists on recognizing articulation parameters of SL articulators. This involves finding articulators on corpora and determining their states; knowing where they are positioned, which configurations they hold or whether they are active in discourse.

Individual sign recognition Classification of individual signs, both on isolated and continuous recognition tasks. Research in this area also includes sign segmentation: finding time-intervals corresponding to meaningful productions on corpora, regardless of the specific signs depicted by the material.

— Isolated sign classification: Identification of single signs — mostly lexical — on idealized conditions.

— Continuous sign classification: Recognition of signs during discourse, where a lot of uncertainties and natural variations may present themselves during production.

As implied, sign recognition relies on the correct identification of measurable parameters, which can be a very specialized task. Because of this, emphasis is usually given to techniques centered around SL phonetics, expecting to be able to build from there, in an effort to search meaning from the bottom-up.

3.3.1 Sub-lexical recognition

Sub-lexical recognition is rich in methods to solve different kinds of problems, starting from the identification of individual articulators on corpora. As previously stated, each articulator has its own set of characteristics, and they can be very different from one another. Accordingly, they contribute their own particular kind of linguistic information, which can be different from the others both in terms of form and function. In addition to these differences, corpus media itself may come with their own recognition-related problematics; recall that corpora are compilations of video or motion capture information, which may have an assorted list of shortcomings such like occlusions, irregular forms, signal noise, etc. All these factors can impact on automatic articulation parameter analysis and, in most cases, they are challenging enough to claim the research focus themselves, individually [Ding 2009]. As a result, it is not uncommon for each articulator to have its own set of specialized recognition techniques, without considering how these may communicate in higher levels to actually apply NLP algorithms.

3.3.1.1 Individual articulators and single parameter recognition

One of the first challenges of SL recognition, is to robustly follow articulators under changing conditions. An *articulator tracker*, is an algorithm that calculates the geometrical position of one or various articulators over a given time interval. A lot of the existing trackers are oriented towards the hands, in part because they are featured prominently in corpora; because of their fairly independent nature, hands can be very accurately tracked, as it's easier to differentiate them in video from more nuanced articulations [Vogler 2007]. In spite of this huge advantage, problems may arise on their tracking depending on the specific conditions in which the algorithm is executed, which are not always optimal [Gonzalez 2011].

High accuracy hand-tracking methods like [Gianni 2009, Gonzalez 2011], use *particle filters* [Isard 1998] to classify signers' hands and head with minimal supervision. Their

respective algorithms are designed to work over generic video sequences, acknowledging the that most existing resources are unannotated video corpora (see Chapter 1, Subsection 1.1.1, p. 3). In general, the authors use the skin tone to select the image regions (*blobs*) that could most likely be the hands or head. Each blob is sampled into a "cloud" of particles, where each particle has an associated weight corresponding to the probability of it being part of the articulator of interest. The filter is dynamic; at each step, it takes in account the previous, current and next observations to recalculate particle weight. With enough executions, the algorithm will inductively converge towards the desired model's solution. This enables following hands and head with some robustness, although it might be costly in terms of calculation time. Finally, both methods apply different techniques to compensate occlusion with some success, one relying on using a probability exclusion principle [Gianni 2009] and the other by calculating differences in luminance [Gonzalez 2011]. Nevertheless, approaches relying on skin tone tend to fail under poor illumination conditions, skin-toned clothing, cluttered backgrounds or other sources of noise.

To avoid these potential pitfalls, some authors have approached the segmentation problem by constraining the initial setup of corpora [Starner 1998]; that is, they force very controlled conditions to minimize tracking errors. Other alternatives showcase the use of special markers during the segmentation phase, in order to avoid the use of skin tone; for instance, authors [Dorner 1994, Starner 1995, Grobel 1997] use unambiguously colored gloves to improve results. Other popular techniques rely on the use of specialized equipment such as wired data-gloves and accelerometers [Kadous 1996, Fang 2001], sometimes also paired with cameras [Brashear 2003], which can obtain accurate tracking data by design. Works like the one presented by [Lu 2010] use state-of-the-art motion capture technology to obtain tracking information of hands, body, head and even eye motion. Finally, some corpora have been created with 3D motion sensing devices like the Kinect [Zafrulla 2011, Stefanov 2013], which can give enough information to find the hands and head without markings.

Novell unmarked methods like [Buehler 2008], try to also avoid specialized equipment. In their work, the authors use a geometrical model to determine where the hands and head should be, based on the form of the human body. Broadly, a geometrical model of the human body is overlaid into each frame. From there, an algorithm calculates the probability of blobs of similar color of being part of any of six body parts defined by the model (hand, arm, shoulder, etc.); blobs are determined by a measure of color proximity, not linked to any particular tone. Based on this process blobs obtain a label, which is checked dynamically against previous calculations to improve accuracy. A different proximity measure is obtained between images, in order to optimize comparisons; in very similar images, the algorithm will check only the subset of information that has

changed. The continuous application of the algorithm, permits inferring position changes with high accuracy. The main advantage of this method with respect to others is its overall robustness, as it can be applied to treat videos created under very different conditions [Buehler 2009].

Articulator classification and hand position tracking serve as the basis for more complex treatment. It is at this point that the difficulty of the task increases, as it is necessary to automatically find other less evident parameters. From these, the detection of hand-configurations has gained traction over the years; however, it has proven to be one of the most challenging. This is mostly due to the deformable nature of the hands [Vogler 2004], which can complicate the task. In addition, variations between configurations can be minimal, to the point of being nearly indistinguishable: for example, the configurations shown in Figure 3.15 only differ on the flexion of the index finger and thumb, which is enough to mark a difference between the two.

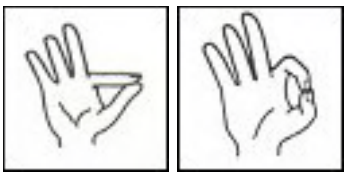


FIGURE 3.15: Two very close LSF hand configurations. [10]

On video, configurations like these can prove to be difficult to differentiate, specially with corpora captured under poor conditions. From the point of view of recognition, this may result into very ineffective classification algorithms with a large number of very closely related classes. This situation also comes with increased computation costs, as the selected classificators have to be thoroughly trained to measure the high quantity of possible small variations. The same situation can present itself with other parameters presenting similar characteristics, such as NMFs [Vogler 2007].

Authors like [Erenshteyn 1996] have tried to address these problems by creating a classification hierarchy for SL video-frames, prioritizing some classificators with respect to the others. In this view, classes are subdivided into small groups of simple classificators. Also, each classificator can be part of more than one classes. To determine its class, each sample is tested by each classificator, and the results are combined into a single score. Classificators are tested following their order of importance on the classification hierarchy. After each pass, the score is compared against an *optimality criterion*, to determine if further tests are required or if the sample can be classified within a certain confidence interval. In this manner, only a subset of the possible classificators is always tested, improving classification times overall.

10. Images obtained from [Companys 2004].

On the same vein, efforts like the one presented by [Munib 2007] use *neural networks* [Hagan 2002] to achieve classification over unmarked images. This method is based on specializing sets of neurons to recognize minimal properties present on the parameters. Thereafter, the topology of the network can be used to combine the results of individual neurons, essentially following the same strategy as the previous example. However, in both cases having reliable training data is essential, and the quality of the input images can always become an issue. For this reason, similar to what happens with position tracking, several research projects rely on sensors to improve the quality of data from capture [Takahashi 1991, Kadous 1996, Liang 1998], which improves accuracy and reduces calculation times.

3.3.1.2 Segmental feature recognition

Some research projects avoid the recognition of isolated parameters altogether, and treat the problem from a morphologic perspective; this is, they classify entire postures, trying to identify sign components instead of individual parametric variations. One of the main advantages of these techniques, is that additional linguistic information can be used to improve recognition conditions; for instance, it may be easier to discard noisy video segments by catching meaningless postures [Gonzalez 2012], or to isolate important gestures [Nam 1996] such as pointing [Nickel 2003]. It turn, this can also help to develop higher level recognition tasks, which could also benefit from SL sequential characteristics to improve accuracy [Vogler 1997].

A lot of the projects developed under these assumptions use Hidden Markov Models (HMMs) [Stamp 2004] for their machine learning tasks; roughly, a HMM represents a probability distribution over a sequence of independent observations of some process. Usually, HMMs are represented as graphs, where each transition is labeled with the independent probability of changing from one state to another. The model is based on the assumption that there are states *hidden* from the observer, and these can be inferred from observations so as to predict the future of the represented process. Due to their flexibility, they are frequently used to model time series data such as video recognition, computational biology, data compression and others.

Works based on classic HMMs and the like usually avoid tracking, as they intend to work directly over unmarked corpus images. For example, the authors [Cooper 2007] encode simultaneous parametric information into two independent classes: static features and movements. The system then analyzes raw video data, and classifies frame sequences into one of the two classes. Each found sequence is characterized by its own *feature vector*, regardless of the class it was classified into. To prove the utility of their approach, a

lexical sign database was created describing each sign's formation; as with video sections, each of the sign's segments are characterized by a vector, enabling the development of distance measures to determine how close detected postures are from predefined ones. Finally, these measures can be used as input for morphologic *Markov chain*, enabling sign recognition.

Despite its advantages, the usefulness of the aforementioned method is limited by the quantity of training samples available (see Chapter 1, Subsection 1.1.2.2, p. 9). In addition, linguistic variations and the need to introduce more or less parameters to the model forces the creation of several HMMs for the same segment, limiting scalability.

A similar technique was used by [Brashear 2003], where the authors describe individual signs by way of HMMs. In general, each signs' internal structure is implicitly captured by its corresponding HMM. This implies that longer signs will thus induce larger models. Similarly as what happens with previous efforts, the lack of learning data limits both the creation and evaluation of the resulting models, forcing researchers to test their results over rather small sets of signs.

Works like [Vogler 2001, Vogler 2004] have tried to improve scalability by considering modalities as independent channels, on their own variation of HMMs. As a result, the system can do dynamic addition and removal of parameters to the model during recognition, which both improves the framework's flexibility and performance. In some ways, their approach works as a mixture of posture-based and parameter-based recognition, as this makes possible the definition of single parameter lexemes, among others. The authors base their framework on the phonological model by [Liddell 1989]; in this sense, the use of knowledge on sign structure helps them reduce computation times, by exploiting techniques reminiscent of hierarchical classification.

Approaches like the ones tried by [Ma 2000, Liu 2014], attempt to integrate NMFs to the recognition process, as a way to make sense of data as a whole rather than concentrating on individual parameters.

Other projects generalize the same principle, in an effort to obtain more information about a posture's intention; for example, [Green 2004] present an architecture to track human body movements, for biometric analysis. In the article, the authors use both a particle filter and HMM to reduce computation times and improve accuracy. Articles like [Bauer 2002b], explore the idea considering variations on technology, and the general problem of subdividing SLs signs in segments. Finally, projects like [Gonzalez 2012] explore the use of low-level tracking-data to calculate automatically a proposed sign segmentation, in an effort to remove co-articulations from video data in annotation tasks.

3.3.2 Individual sign recognition

In this kind of recognition tasks, it is common to have two distinct classification phases:

1. sub-lexical recognition, either of individual parameters or segmental features;
2. sign classification.

Due to the bottom-up approach adopted by the domain, the hit and miss ratio of sub-lexical feature recognition from the first phase will necessarily impact the accuracy of sign classification as a whole [Waldron 1995, Brashear 2003, Cooper 2007].

The application of this type of hierarchical philosophy, inevitably limits the capabilities of recognition tasks from a technological point of view: each level could theoretically impose new restrictions on data, which in turn could reduce the amount of exploitable information. For example, calculation-heavy parameter recognition based on particle filtering [Lefebvre-Albaret 2010, Gonzalez 2011], may not escalate well for real-time processing, which would immediately discard some of the existing data sources, developed to be treated under this technique. In addition, this approach is also prone to suffer from cascading failures, carrying erroneous data from lower to higher levels, and limiting usability. Both these technological limits, and the existence of different research objectives, has separated the field into two distinct areas: *isolated* and *continuous* sign recognition.

As its name implies, isolated sign recognition studies how to recognize individual apparition of signs [Grobel 1997]; in general, research in the area centers around the identification of single signs in very well-delimited contexts. This could be useful in different situations such as Deaf machine interaction [Heloir 2015], the search of signs by signing on visual data repositories [Elliott 2012], the creation of software solutions for SL education [Stefanov 2013], and even to enable simple signing access to new technologies [Fotinea 2012]. Most of the aforementioned projects, rely on the study of specific use-cases, where sign delimitation is almost always a given, and where the quantity of possible signs to recognize is limited.

On the other hand, continuous recognition tries to identify sign apparitions over unbounded contexts. This can be useful for projects related to the identification of phrases [Chai 2013], discourse analysis [Bauer 2002a, Nayak 2009], or for the recognition of complex linguistic phenomena [Kelly 2009].

Contrary to what happens on isolation, continuous recognition has to work over very unstable environments, where the system is obliged to independently find cues leading to the segmental subdivision of the utterance [Bauer 2002b]. Notably, most of the existing projects on the area work by chaining HMMs; that is, they first create smaller HMMs

for individual sign identification, and the result serves as input for a morpheme-level HMMs [Ong 2005]. However, these kind of approaches can be difficult to develop, because of the lack of both enough training data or linguistic knowledge to create the models. These limitations, alongside the hierarchical structure, end up translating into very reduced sets of recognizable signs, as exemplified by Table 3.1.

Authors	Type of Recognition	Technique	Qty. of Signs
[Waldron 1995]	Isolated	Neural networks	14 signs
[Vogler 2001]	Continuous	HMMs	22 signs
[Bauer 2002a]	Continuous	HMMs	97 signs
[Brashear 2003]	Isolated	HMMs	5 signs forming 72 phrases
[Cooper 2007]	Isolated	Markov chains	164 signs
[Kelly 2009]	Continuous	HMMs	8 signs, 100 co-articulations

Table 3.1: Quantity of recognizable signs by research project.

For comparison, projects in continuous speech recognition are capable of recognizing sets of more than 1000 words using similar techniques [Roth 1993], even in very limited conditions. This is of particular interest for the domain, as the existing data resources are not only limited but very heterogeneous, as it will be explained in the next section.

3.3.3 Notable Sign Language Research Resources

Various corpora for research have been developed over the years, with special emphasis on linguistics rather than computer-friendliness. One of the main characteristics of these corpora, is that they try to capture as many different variants of the language as possible. In general, this differs from the current approaches used by computer-based SL research, where ideal "standardized" language samples have been preferred [Morrissey 2010, Forster 2012]. This is because, as mentioned in previous sections, the existence of variation can affect evaluation results on both synthesis and recognition tasks. As such, classic linguistic corpora pose different challenges to researchers who wish to use them as data sources, as they depict data in far from ideal conditions for automatic processing. Regardless, they remain prominent data sources for computer-based research, due to their relatively high availability.

For example, [Crasborn 2008c] present the authors' efforts to record a corpus for the Sign Language of the Netherlands (NGT). For their project, the research team collected samples from 100 native signers, coming from different backgrounds and regions. The participants were tasked with signing narratives, using popular culture bits as *elicitation*

materials (cartoons, comic stories, TV videos, etc). Other tasks included discussions around SL issues and even free conversation. From a technological point of view, the corpus was captured using four High Definition cameras, located around the signers as shown on Figure 3.16.

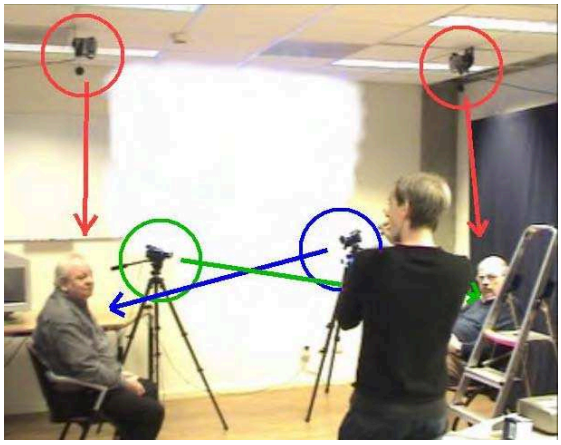


FIGURE 3.16: Recording setup for the NGT corpus [Crasborn 2008c].

The project resulted in around 80 hours of video data, available for its consultation over the web. Some annotations and translations were made for some of the videos, however budget and time limitations didn't let researchers annotate everything nor give very detailed glosses. In any case, the work resulted in the development of new software for on-line collaboration [Brugman 2004], and on the integration of new annotation capabilities to the ELAN platform [Crasborn 2008a].

It is common for long-term corpus projects to develop their own capture standards, which can vary depending on the size and scope of the project. Because of this, corpora has been built upon tools as diverse as professional video cameras [Hanke 2010, Schembri 2013], motion capture equipment [Cox 2002, Lu 2010], televised SL transmissions [Buehler 2009], or even with new emerging hardware technologies [Stefanov 2013]. Similarly, these projects frequently develop their own annotation tools, which usually implement their own standards. As such, big corpus projects have influenced the development of new annotation formats, as it is common for their auxiliary tools to become available to the research community. This availability has influenced the design principles of new, smaller corpora; for instance, authors [Karpouzis 2007, p. 6] express their choice of HamNoSys annotations, because of their compatibility with the ELAN platform.

One of the biggest SL linguistic data repositories to date is the DSG Corpus Project [Hanke 2010]. For its creation, 300 signers from around 12 different regions were selected, in an effort to capture as much linguistic variations as possible. Being a long-term project, DSG managed to amass over 500 hours of video samples, showing approximately 3 million

different signs. However, only a fraction of the materials are freely available [Nishio 2010], and not all videos are annotated; only 18 sessions out of 3 regions have annotations. An specialized software platform called iLex [Hanke 2002] was used for its annotation. In particular, this tool permits the expression of annotations as either glosses or as HamNoSys descriptions.

With respect to capture, the corpus was recorded by pairs of signers with the help of three HD cameras: two to capture each signers' frontal plane, and the last one recording the complete scene. Figure 3.17 shows a shot of the camera setup.

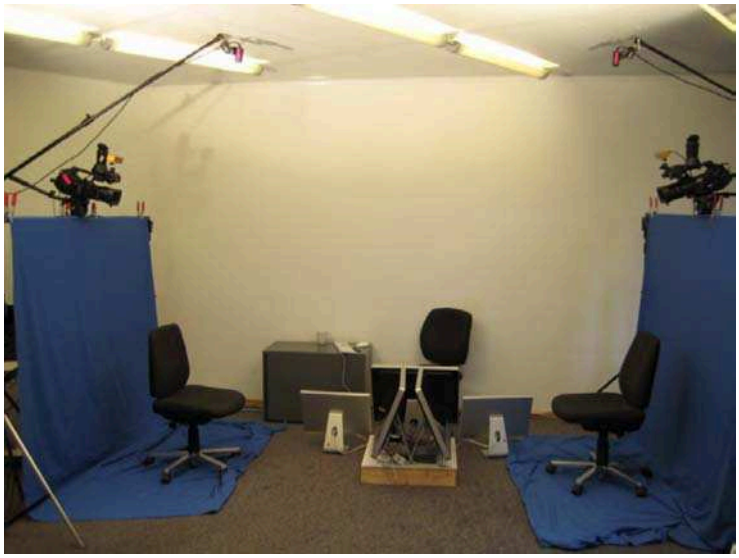


FIGURE 3.17: Example of DSG and Dicta-Sign setup [Hanke 2010].

This same setup was used for the Dicta-Sign corpus, presented by [Matthes 2012]. This project consisted in creating a compilation of annotated corpora from four different European SLs (BSL, DSG, GSL and LSF). Data of the four languages was recorded in parallel, using the same elicitation materials and a lexicon of more than 1000 signs. Despite of the corpora being elicited to reflect common use of SLs, Dicta-Sign was created with the intention of serving as input data to develop computer-based research. For this reason, special emphasis was put on creating computer readable annotations, that could be used to interpret data automatically.

Elicitation tasks for each corpus were performed by groups of 16 native signers, paired together into sessions of around two hours [Hanke 2010, Fotinea 2012, Matthes 2012]. From the resulting raw video, at least 5 hours of lexical gloss annotations were created, summing up to 25 hours of annotated corpus-data. Annotations were created with the help of the iLex platform, following the same guidelines across the four corpora. Some variations in annotation do exist, however, due to personal choices of the teams creating each corpus.

As with other projects of similar size, new software tools oriented towards annotation were created as part of the project. Contrary to previously existing software, the approach taken by Dicta-Sign relies on the integration of classic multimodal annotators with specialized SL software modules, following the guidelines introduced by [Braffort 2004]. The main idea is that, by way of these modules, human annotators can collaborate remotely [Collet 2010, Dubot 2012], or can introduce recognition capabilities that could potentially simplify the annotation task [Gonzalez 2012].

Finally, the Dicta-Sign project resulted in the development of new corpus-based research, showcased by the production of several proofs-of-concept in the areas of web communication for the Deaf [Efthimiou 2012b], SL synthesis applied to human-computer interfaces [Efthimiou 2012a] and isolated sign recognition on lexical databases [Elliott 2012].

Similar long-term non-European efforts are represented by projects like the National Center for Sign Language and Gesture Resources (NCSLGR) corpus of ASL [Neidle 2000]. For their work, the researchers captured a database of 1866 distinct signs, performed by four native signers, summing up to 11,854 total sign occurrences on video. Capture was achieved by way of four standard definition cameras (no HD), positioned to capture orthogonal planes of the participants. An example is shown in figure 3.18.

FIGURE 3.18: Example of the different angles captured for the NCSLGR corpus [Neidle 2000].

The corpus was annotated with SignStream, an specialized SL annotation tool designed specially for the project [Neidle 2001]. Even though the corpus wasn't originally intended to be used in computer-based research, further annotation iterations have resulted in the development of computer-friendly distribution formats; for instance, SignStream is able to export annotations in both its own well-defined standard and as XML files. This evolution has enabled the use of their corpus in several recognition tasks across more than a decade [Zahedi 2005, Neidle 2009, Wang 2012].

In a final note, smaller corpus projects are starting to take advantage of the byproducts of larger ventures; researchers in linguistics begin to acknowledge the importance of computer-based tools for corpus analysis, even in purely linguistic research [Leeson 2006]. Indeed, the literature shows that the vast majority of new linguistic SL corpora are systematically subject to some kind of automatic treatment. By the same token, corpora created exclusively for data-driven research have started delving deeper into linguistics [Forster 2014], as linguistic knowledge is an important factor for the correctly interpretation of data. For these reasons, the development of new, more integrated, approaches towards corpus representation and annotation seem unavoidable, as it has become more evident than ever that neither purely mechanical nor purely linguistic aspects are sufficient on their own to understand SLs. Fortunately, computer science already fosters a set of procedures designed to treat similar issues, as they were created to match very technical problem descriptions with higher-level abstractions. In the literature, these techniques are usually referred as *formal methods*.

3.4 Formal Methods

The term *formal methods*, encompasses a group of mathematical techniques used to describe and test software or hardware systems, from conception to implementation [Wing 1990]. These are usually subdivided into two complementary areas:

Formal Specification Contains a series of formal notations used to characterize systems (specify), harnessing the power of several mathematical models.

Formal Verification It refers to the series of techniques that serve to test (verify) formal specifications and their implementations, to assure that they conserve the desired properties.

Specification and verification are considered important tools to avoid pitfalls in the creation of complex systems; by introducing mathematical rigor, system designers can build coherent and unambiguous descriptions, which are both easier to understand and less likely to fail [Bowen 1993]. However, despite their benefits, their penetration outside of the domain of critical systems remains lackluster at best, specially because of hurdles related to the both the usability of formal notations and other pragmatical issues [Knight 1997]. Regardless, some readily available solutions exists and have been central to the development of complex industrial processes in areas such like transportation [Hall 1996], micro-processor design [Bentley 2001] and energy production [Yoo 2008]. In the next sections, some of the existing work in the area will be presented, in an interest of showing the flexibility of formal notations and their applicability to various areas.

3.4.1 Formal Verification

The process of *formal verification* entails the analysis of a system's design, in order to determine if it satisfies certain properties [Baier 2008]. This process can only take place if the system is defined formally; that is, if there is no ambiguity on its definition, and if there exist a well-defined set of consistent rules framing its inner workings. This makes the use of formal notations specially suitable for verification, which usually serves as a compelling argument to advocate for their use during system design [Holloway 1997].

There are two main techniques commonly used for verification: *model-checking* and *automated theorem proving*. The choice of one instead of the other depends largely on the specification formalism selected by the designer; however, mathematically speaking, both methods are equivalent [Clarke 1986]. That said, model-checking is better adapted to verify systems represented as Labeled Transition Systems (LTSs) [Keller 1976, p. 372], whereas theorem proving methods tend to be favored for the verification of more symbolic representations [Chang 1997].

In general, model-checking algorithms are based on performing exhaustive searches for every possible state of a LTS, in an effort to find all possible errors on the system's specification [Emerson 1997]. However, this methodology comes with the serious disadvantage that it has to deal with combinatorial explosion. Accordingly, model-checking algorithms are limited by tractability issues; this introduces the problem of determining if the verification of some description will be intractable, even before its creation. For this reason, research in model-checking has been focused on finding the limits of exhaustive verification [Bauland 2009] and developing new stochastic methods to optimize performance even in the worst cases [Kwiatkowska 2007].

On the other side of the spectrum, automated theorem provers try to tackle intractability by avoiding the systematic search of every possible state. To this end, these tools make use of different *proof calculi* to determine if a specification is valid, obtaining proofs by the strategic application of a finite set of inference rules [Chang 1997]. Specifications verified under this method, usually have to provide both a set of axioms modeling the system's invariants and a set of properties to prove; from there, the inference engine will try to find a formal proof asserting to show that the system fulfills each property, in order to determine the correctness of the specification.

Theorem proving is limited by decidability issues; for most formal systems, no algorithm exist to prove that some statement is true or false in the system [Davis 1958]. For this reason, some of the existent research on formal methods deals with proving the decidability of description formalisms; in practice, this results in the creation of less expressive descriptions, albeit automatically verifiable.

3.4.2 Formal Specification

A *formal specification* is a mathematical description created to capture, unambiguously, the inner workings of a system. They are written by way of very specialized notation languages, emanated from within various disciplines of mathematics.

Traditionally, formal notations have been inspired on logic, graph theory or algebra, depending on their intended use. In practice, the choice of one formalisms with respect to the others corresponds to the description needs of the system's designer; in this regard, the choice of which formalism to use and how, becomes part of the specifier's tasks, mostly depending on the problem and the purpose of the specification. For instance, in software development, designers can either use Z-notation [Woodcock 1996] to characterize the interaction of an entire system, or they can go with Larch [Guttag 1993] to describe on an abstract level how the most basic components work. As such, it is not uncommon for specification languages to share similar expressive capabilities, since they have to show some degree of flexibility in order to adjust to the different possible use-cases.

Formal specification notations have been traditionally grouped into classes, semantically related to their underlying mathematical basis; the most relevant, for this scope of thesis, are listed below:

— model-based;
— algebraic;
— process-oriented.

For the most part, this classification is fairly subjective, as some formalisms could fit in more than one category. In the next sections, some examples of each listed type will be analyzed, as well as their overall relevance in terms of this work.

3.4.2.1 Model-based specification

Model-based specifications are built upon concepts of set theory [Jones 1980]. As such, notations in this category will let designers describe information in terms of sets and relations between them; normally, operations are defined as mappings from one set into another. System invariants or semantic rules linked to set definitions, are usually described with predicates of first order logic. This enables the depiction of systems as complex mathematical structures, like LTSs. An example of this can be seen in figure Figure 3.19, built on Z-notation [Woodcock 1996] and taken from [Jacky 1996].

STATE ::= patients | fields | setup | ready | beam_on
EVENT ::= select_patient | select_field | enter | start | stop | ok | intlk
LTS == (STATE × EVENT) ↦ STATE

no_change, transitions, control: LTS
control = no_change ⊕ transitions no_change = { s: STATE; e: EVENT • (s, e) ↦ s } transitions = {(patients, enter) ↦ fields, (fields, select_patient) ↦ patients, (fields, enter) ↦ setup, (setup, select_patient) ↦ patients, (setup, select_field) ↦ fields, (setup, ok) ↦ ready, (ready, select_patient) ↦ patients, (ready, select_field) ↦ fields, (ready, start) ↦ beam_on, (ready, intlk) ↦ setup, (beam_on, stop) ↦ ready, (beam_on, intlk) ↦ setup }

FIGURE 3.19: An schema in Z-notation representing a system for patient control in a hospital.

The shown specification defines a set of states STATE = {patients, fields, setup, ready, beam_on}, a set of events EVENT and uses them to describe the system as the map LTS : STATE × EVENT ↦ STATE, taking pairings of states and events into new states. Thereafter, the description goes into detail, describing which pairs correspond to which states in the actual system. This information can be represented graphically by the diagram in Figure 3.20, showing the possible states of a medical therapy control system.

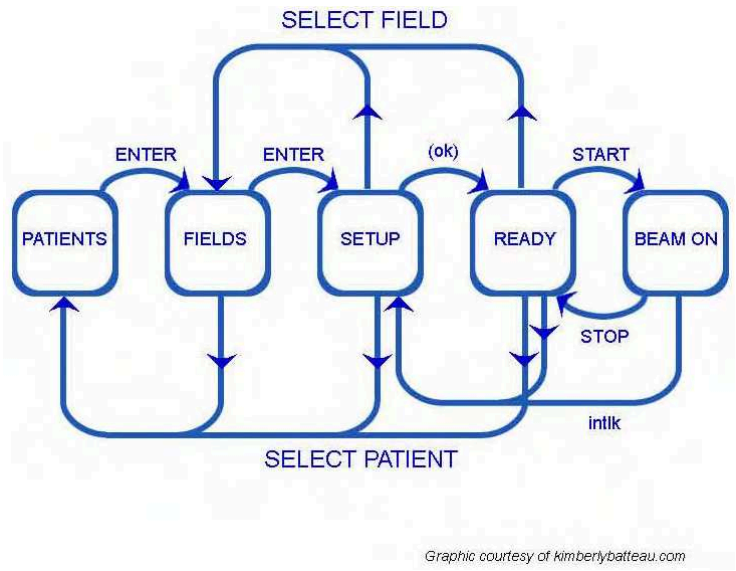


FIGURE 3.20: LTS representing a therapy control system.

This kind of description specifying system functions as maps between sets, can be useful to verify implementations, independently of the technical details behind each state or transition. A high level verification tool could, for example, check each operation by

asserting that their results are members of the appropriate sets. In this way, it is possible to isolate the model from its implementations, enabling verification for all of them.

Other popular formalism relying on the same principles is the Vienna Development Method (VDM) [Jones 1990]. The VDM specification language uses *types* as primitives, which are none other than sets over specific domains. Figure 3.21 shows a definition of types on VDM.

```
types

Country = seq of char;
Relation = set of (Country * Country);
Colour = set of Country;
Colouring = set of Colour;
```

FIGURE 3.21: Type definitions in VDM specification language for a colored world map.

In the example, `Country` is defined as the set comprising any possible sequence of characters, depicting the name of a country. A relation between countries is defined as the set containing every possible pairing of sequences of characters in `Country`, `Relation` = `Country` × `Country`. Also, like in Z-notation, sets can be combined into more complex types to define operations. An example of a function's definition is shown in Figure 3.22.

```
functions

sameColour: Country * Country * Colouring -> bool
sameColour(cn1,cn2,cols) ==
exists col in set cols & cn1 in set col and cn2 in set col;
```

FIGURE 3.22: Function definitions in VDM specification language.

Both the Z-notation and VDM have been readily used in industrial applications with success [Craigen 1993], and continue their evolution with respect to their use-cases in industry; however, other less industry-friendly theoretic formalisms are abound, exploring similar issues from a rather symbolic point of view. Such is the case of algebraic notations.

3.4.2.2 Algebraic specification

Algebraic specification notations represent systems in a more basic level than previously reviewed formalisms; they represent the inner workings of a system as algebraic structures. This is done by way of abstractions like the ones used by VDM, where every piece of

data has an associated Abstract Data Type (ADT), defined only over a specific set of values. However, unlike model-theoretic specification languages, algebraic notations are designed to express systems in more general terms, semantically closer to mathematics than to the actual implementation [Guttag 1978].

ADTs are composed of two main parts:

— The signature of each one of its operations, defined recursively in terms of the given types.
— A set of axioms describing the semantics of these operators.

An example of an ADT is presented in Figures 3.23 and 3.24, where an algebraic specification of a Set of integers is given.

$$\begin{array}{rcl} \text{EMPTY\_SET} & : & \rightarrow \text{Set} \\ \text{INSERT} & : & \text{Set} \times \text{Integer} \rightarrow \text{Set} \\ \text{DELETE} & : & \text{Set} \times \text{Integer} \rightarrow \text{Set} \\ \text{MEMBER\_OF?} & : & \text{Set} \times \text{Integer} \rightarrow \text{Boolean} \end{array}$$

FIGURE 3.23: Definition of a Set of integers as abstract data types.

In general, just as with VDM, operators are defined in terms of maps between sets. Figure 3.23 gives the signatures of an empty set (mapping $\{\epsilon\}$ to a Set), operators closed under Set for insertions and deletions, and the signature of an operator which determines if an integer is member of a Set. The axioms defining the semantics are given in Figure 3.24.

1. MEMBER_OF? (EMPTY_SET, i)=false
2. MEMBER_OF? (INSERT(s, i), i')=**if**?=?(i, i')
   then true
   else MEMBER_OF?(s, i')
3. DELETE (EMPTY_SET, i)=EMPTY_SET
4. DELETE (INSERT(s, i), i')=**if**?=?(i, i')
   then DELETE(s, i)
   else INSERT(DELETE(s, i'), i)

FIGURE 3.24: Axioms for the definition of the Set of integers.

The axiomatic specification serves as high level abstraction of what it means, in terms of set transformations, to insert, delete or be a member of Set.

The flexibility of ADTs and their symbolic description approach, has permitted the creation of the Larch notation [Guttag 1993], which can be tailored to interface with different programming languages and to specify very specialized behaviors, such as the ones found in concurrent systems [Lamport 1989].

3.4.2.3 Process-oriented specification

Process-oriented specifications are mainly focused on how a system reacts. As such, these notations usually give the designers the means to create abstract descriptions of a system's evolution [Winstanley 1991]. Notably, the interpretation of process-oriented descriptions is usually done over LTSs.

A lot of the process-oriented formalisms are used to specify concurrent systems; for this reason, most of the existing notations have evolved to be able to represent characteristics such as message-passing and parallelism [Pierce 1995]. It is also common for this kind of tools to formalize concepts such as channels, and to define groups of indivisible communication primitives that the processes can execute non-deterministically.

Notations like Calculus of Communicating Systems (CCS), developed by [Milner 1982], captures the behavior of concurrent systems by combining processes recursively as defined by the Backus Normal Form (BNF) shown below:

Definition 3.1 (Process in CCS)**.**

$$P ::= \emptyset \mid a.P_1 \mid A \mid P_1 + P_2 \mid P_1|P_2 \mid P_1[b/a] \mid P_1\backslash a$$

where:

- $\emptyset$ denotes the empty process.
- $a.P_1$ indicates that a process executes action a and continues as P_1
- A is an identifier representing a primitive process.
- $P_1 + P_2$ means that either P_1 or P_2 continues.
- $P_1|P_2$ indicates that processes P_1 and P_2 exist simultaneously.
- $P_1[b/a]$ denotes the process P_1 where all actions a are renamed as b.
- $P_1\backslash a$ Represents the process P_1 without action a.

■

In this specification language, actions can represent any possible communication primitive needed by the designer to describe the system, and are defined depending on the specific use-case.

Communicating Sequential Processes (CSP) is another formalism working under similar assumptions. Originally presented by [Hoare 1985], CSP follows the same basic model than CCS: it describes primitive *events* and processes to represent the particulars of a system's functionalities. From there, the description of more complex functions is done recursively, by way of a the group of logic operators presented below.

Definition 3.2 (Algebraic operators of CSP)**.**

$$P ::= a \rightarrow P \mid P \Box Q \mid P \sqcap Q \mid P|||Q \mid P||\{a\}||Q \mid P \backslash \{a\}$$

where:

— $a \rightarrow P$ is the process which, after event a, behaves as P.
— $P \Box Q$ denotes the process where either P or Q are executed.
— $P \sqcap Q$ denotes the process where either P or Q are executed, non-deterministically.
— $P|||Q$ is an interleaving process, where P and Q exist simultaneously.
— $P||\{a\}||Q$ is a synchronized interleaving, where a must occur in both P and Q before parallel execution.
— $P \backslash \{a\}$ is a process resulting of removing event a from P.

■

Finally, the Propositional Dynamic Logic (PDL) captures similar process-oriented properties, centering around sequentiality rather than concurrence. PDL is a multimodal [11] logic [Chellas 1980], first defined by [Fischer 1979] to characterize computer programs. Originally, it provided a formal framework for program description, allowing them to be interpreted as modal operators. The syntax of PDL is given below:

Definition 3.3 (Syntax of PDL)**.** Let α denote a program and φ a PDL formula.

$$\alpha ::= \pi \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\alpha]\varphi$$

where:

— π denotes an atomic program.
— p denotes an atomic proposition.
— $\alpha; \alpha$ indicates the sequential execution of two programs.
— $\alpha \cup \alpha$ is the non-deterministic choice between two programs.
— α^* means that program α is executed a finite, non-deterministic number of times.
— $\varphi?$ is the program that fails if φ is false.
— $[\alpha]\varphi$ indicates that, after executing program α, the system necessarily enters into a state where φ is true.

■

11. The term *modality* has different meanings in semiotics and formal logic. While in the former it has been previously defined as akin to a communication channel (see p. 28), in the latter it refers to the qualification of a statement in terms of "necessary" and "contingent" truths. In this regard, a logic is multimodal if more than one qualifiers are available.

In PDL, system properties are described as formulae and, as it's common with other modal logics, its semantics are interpreted by way of LTSs.

Several variations of the logic exist, with different repercussions over its decidability. However, the basic version presented in Definition 3.3 features an optimal model-checking algorithm [Lange 2006], and it has been proved decidable [Danecki 1985].

Other process-oriented formalisms are explored in detail on the literature [Winskel 1995], where different tools have been developed to enable specification and automatic verification of concurrent systems. In this regard, the area has provides a basis to attempt the formalization of SL production rules as transition systems: it has developed specification languages, capable of expressing simultaneous behaviors between independent agents and their temporal properties. These are desirable characteristics for the modeling of SL morphology, in particular when depicting how articulators act independently but, nevertheless, synchronize to generate meaning. In addition, these notations are designed around their *expressiveness* — their capacity to represent information — and, thus, are broad enough to describe SL utterances in different situations. In this way, they are capable of producing linguistic annotations which can be enhanced with technical information about any given corpus, something that can be useful in the automatic analysis of SL corpora, as presented in the next chapters.

3.5 Chapter conclusion

In this chapter, some of the most common SL representation formalisms were introduced, alongside some of their applications in research. In addition, a brief overview on SL synthesis and recognition was given, as well as a general introduction to formal methods.

As shown on the chapter, some of the most common problems in research and representation, come from SLs multimodal nature. This not only increases the complexity of synthesis and recognition tasks, but also affects the availability of copora due to the variety of technologies used to deal with multimodality. This prompts for the search of new representation methods that could help unify the current heterogeneous environment under a common meta-language, friendly not only for linguistics but also for very technical tasks.

It is on this context, that this thesis presents its main contributions. The idea, was to develop a flexible phonetic representation framework, capable of specifying articulation parameters in a way similar to what SiGML and the Stokoe notation do. Additionally, even though the proposed framework is strictly phonetic, it was built under the assumption that it would eventually be extended to represent morphological SL properties. With

this in mind, the framework provides a broad set of tools intended to go beyond phonetic description in the future, akin to what Zebedee has done with AZee [Filhol 2012b]; the main assumption being that a flexible phonetic basis, can lead to higher level descriptions.

In the next chapter, a new SL phonetic representation method is presented, based on the principles of formal methods; even though specification notations have been heavily influenced by industrial applications, the same principles can be applied to academic research. For instance, linguistics have a rich history of developing formalisms for the study of natural language [Chomsky 1957,Pullum 2005,Moss 2007], which has contributed on the theoretical development of the area. In particular, the proposed formalism is modeled on the PDL, also reviewed in the chapter, a logic created to represent change.

Chapter 4

Formal Specification of Sign Language Utterances

The previous chapter presented a series of formal specification languages, striving to thoroughly describe complex systems (see Subsection 3.4, p. 67). By way of these notations, system designers can build abstract representations of a system's inner workings, so as to better understand and test its limits. Sign Languages (SLs) lend themselves to the same kind of formalization: they can be seen as complex transition-based systems (see Subsection 2.2.1.4, p. 29), able to produce phonetic change by applying per-articulator rules of change.

Recall from previous chapters that computer-based SL study is plagued by several problems deterring the development of the domain:

— the fact that SL corpora are collections of non-textual data, forcing Natural Language Processing (NLP) algorithms use uncommon data — such as video images — as input;
— the variability of corpus-data, impeding the creation of vast language information repositories such as the ones existing for the study of vocal languages; and,
— the difficulty to annotate these corpora, taking into account both that annotation tasks are also very variable, and that there is a limited availability of SL experts able to create and correct annotations.

In this regard, using formal specification techniques to convert corpora into abstract representations may prove useful, as these abstractions may be used to tackle variability; in addition, they provide a more computer-friendly representation of raw data, which can contribute towards the development of automatic recognition and, crucially, annotation algorithms.

Previous formalization approaches to language study have been developed before [Chomsky 1957, Pullum 2005, Moss 2007], a practice which has also extended to SLs (see Section 3.1, p. 42). However, these efforts continue to be limited by spatial and multimodal representation issues.

With this in mind, this chapter presents a formalism for SL utterance modeling inspired on the Propositional Dynamic Logic (PDL) (see Section 3.4.2.3, p. 74). PDL is a process oriented specification language, centered around sequential modalities. The idea was to extend on this notion to express SL phonetics in similar terms, by linking logical modalities to articulatory actions and depicting how these can foster change. This resulted in the creation of the Propositional Dynamic Logic for Sign Language (PDL_{SL}), which provides a flexible representation of SL multimodality (see Subsection 2.2.1.3, p. 28), paired with a formal notation to express properties over modeled utterances. Together, these tools can help an automatic system to semi-automatically find phonetic properties in a visual corpus, just by ensuring a minimal description of each video by way of the formal structure provided by PDL_{SL}. Globally, the quantity of information that needs to be described is largely decided by the human *annotator*, who ends up fulfilling the role akin to system designer in industrial applications.

The advantage of using PDL_{SL} as a vehicle for the construction of an automatic system, is that it can serve as an intermediate language between computer science and linguistics, minimizing common pitfalls present in other representation languages which tend to favor either one side or the other. For the same reason, this formalism intents to be compatible with other SL phonological representation notations, in the sense that it aims to be capable of capturing at least the same information as the others.

Because of this, PDL_{SL} is defined in somewhat broad terms; its objective is not to represent specific parametric configurations, but rather to give a flexible enough framework to represent as many phonetic combinations as possible, in an effort to remain language and corpus agnostic. Due to the combinatorial nature of SL phonemes, this has to be done by defining everything in terms of sets, akin to the definition of *types* in model-theoretic formalisms (see Subsection 3.4.2.1, p. 69). In particular, this can also help PDL_{SL} to supersede existing transcription notations in automatic tasks, as they are capable of representing (at least) the same information. However, unlike model-theoretic tools, PDL_{SL} types are predefined by the formalism itself, as presented in the following pages.

In the next section, some basic model-theoretic definitions are given to the reader, in order to establish the representation domain of the proposed formalism. Short after, in the following section, the logic's syntax is provided, mimicking the traditional logic-definition mechanism in the literature. Finally, the last section presents the language semantics;

that is, how formulae built with this logic are to be interpreted, related to an abstract SL corpus representation.

4.1 Sign Language Primitives

Like previously mentioned, PDL_{SL} enables the definition of type *primitives* in the same fashion as model-theoretic formalisms, which is helpful to establish limits on what information an annotator can specify; however, instead of forcing the experts to define these types, PDL_{SL} simplifies the task by introducing a series of semantically-motivated set templates, which can be populated by the annotators with the information they deem relevant. The objective of these templates is to provide PDL_{SL} with a powerful enough basis to be, at least, as expressive as other phonetic transcription notation (see Chapter 6, p. 129).

That being said, the set definitions introduced in the next paragraphs are based on existing parametric descriptions of SL phonemes (see Section 3.1, p. 42). In particular, the interest of PDL_{SL} sets is to ascertain well defined concepts (like hand-shapes or locations), as classes containing concrete objects. For this reason, set members will correspond to observable characteristics; for example, the set of hand-shapes will contain every meaningful morphological configuration of a hand. When combined into more complex sets, these primitives serve to define atomic statements, which will in turn operate as seeds of phonetic formal descriptions.

First of all, PDL_{SL} works under the assumption that each articulator is an independent *agent*, inducing individual information channels (see Subsection 2.2.1.3, p. 28) as the ones proposed by [Vogler 2001]; for this reason, it is first necessary to define who these agents are, in terms of the relevant body parts used in SL communication.

Definition 4.1 (Relevant body parts for SL)**.** Let $\mathcal{NAMES}$ be the set containing all possible chains of characters. The set of *relevant body parts* is defined as $\mathcal{BODY} = \{b : b \text{ denotes a body part visible during signing; } b \in \mathcal{NAMES}\}$. ■

Intentionally, the set $\mathcal{BODY}$ has a very broad definition; the relative importance of different body parts may change between SLs, for which it becomes useful to have a very general characterization. As a result, annotators have enough liberty to specify whatever they need, depending on the language properties they are describing. The same can be done for SL articulators, as shown in definition 4.2.

Definition 4.2 (Articulators)**.** Let $\mathcal{ART}$ denote the set of SL articulators, where $\mathcal{ART} = \{a : a \in \mathcal{BODY} \text{ and } a \text{ is capable of articulation}\}$. ■

As with $\mathcal{BODY}$, the articulator set is defined in very general terms, in order to adapt to any possible linguistic characterization that may affect the definition of articulators themselves. This is, $\mathcal{ART}$ will contain anything that annotators consider an articulator. Regardless, more often than not the hands will be in $\mathcal{ART}$, an thus PDL_{SL} assigns specific symbols to denote them: $\mathfrak{R}$ and $\mathfrak{L}$ for the right and left hands, respectively.

In general, articulation is dependent on the morphological characteristics of articulators themselves (see Subsection 2.2.1.1, p. 26). For this reason, PDL_{SL} defines places of articulation in terms of the individual articulators that can use them, as each will have their own articulation capabilities. To this end, it is also useful to have a characterization for the signing space, as sometimes space locations around the signer are used to articulate (see Subsection 2.2.1.1, p. 27).

Definition 4.3 (Space coordinates)**.** Let the set $\Psi_{\mathsf{SL}} \subset \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ contain all possible three-dimensional euclidean coordinates taking the signer's center of mass as origin, such that for every $p \in \Psi_{\mathsf{SL}}$:

— a signer can physically reach point p with any articulator $a \in \mathcal{ART}$;
— the space on p can become empty at any given time; that is, that no body part named by $b \in \mathcal{BODY}$ locates in p.

■

Roughly, the set Ψ_{SL} contains the spatial region depicted on Figure 4.1, which can be activated by signers on discourse. However, this definition is too general, as it also contains regions lacking of linguistic meaning, not used on signing. Definition 4.4 solves this issue, by defining places of articulation under articulatory restrictions.

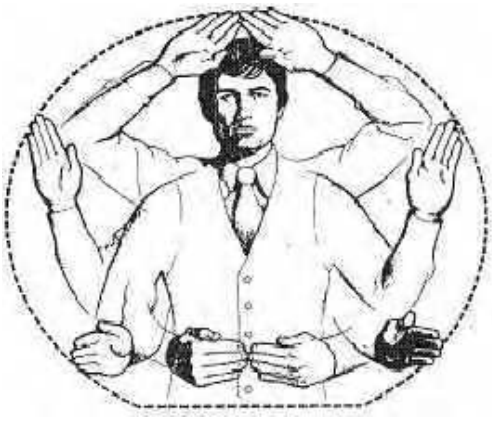


FIGURE 4.1: Spatial region contained by the set Ψ_{SL}.[1]

1. Image obtained from [Venkatagiri 2014].

Definition 4.4 (Places of articulation)**.** Let $a \in \mathcal{ART}$ denote any articulator, and $\mathcal{BODY}^a$ be the set naming any body-bound location where a can articulate during signing, $a \notin \mathcal{BODY}^a$, thus $\mathcal{BODY}^a \subset \mathcal{BODY}$.

Let $\mathcal{C} : 2^{\Psi_{\mathtt{SL}}} \rightarrow \mathcal{NAMES}$ be a function assigning labels to spatial coordinates, and $\Psi^a_{\mathtt{SL}} \subset \Psi_{\mathtt{SL}}$ be the set containing any spatial coordinate where a can articulate during signing. The set $\mathcal{LOC}^a$, denoting *places of articulation for* a, is defined as follows:

$$\mathcal{LOC}^a = \{\lambda : \lambda \in (\mathcal{BODY}^a \ \cup \ \mathcal{C}(\Psi^a_{\mathtt{SL}}))\}$$

Additionally, $\mathcal{LOC}$ is defined as the set containing all possible pairings of valid articulators and articulation places $\mathcal{LOC} = \{(a, \lambda) : a \in \mathcal{ART} \text{ and } \lambda \in \mathcal{LOC}^a\}$. ■

In some cases, Non-Manual Features (NMFs) may change without referring to any external location (see Subsection 2.2.1.2, p. 28). To this end, the proposed formalism permits the definition of partial articulations in terms of configuration change. This is done simply by leaving the location set of the pertinent articulator empty, indicating that its changes will never refer to any location. So, for example, if $a \in \mathcal{ART}$ refers to the eyebrows, then it is most likely the case that $\mathcal{LOC}^a = \emptyset$, as there are almost no instances (if any) where the eyebrows refer to a place of articulation.

It is relevant to mention that these *location-less* partial articulations appear because of the channel separation, induced by the definitions. Intuitively, the formalism models an articulation as a composition of independent articulatory changes, each having the same "value" as a complete articulation. For this reason, to maintain consistency articulation places are described in terms of the articulators that may refer to them.

Similar to what happens with places of articulation, configurations will depend completely on the characteristics of their corresponding articulator: for instance, hand shapes will be exclusively held by the hands, while each NMF will have its own set of valid configurations (*i.e.* raised/lowered eyebrows, open/closed eyes, etc). The definition of configurations under this principle, is given below.

Definition 4.5 (Articulator configurations)**.** Let $\mathcal{CFG}^a$ be the set naming every possible morphological configuration that the articulator denoted by $a \in \mathcal{ART}$ can hold. Notably $\mathcal{CFG}^a \subset \mathcal{NAMES}$ and $\mathcal{CFG}^a \cap \mathcal{BODY} = \emptyset$.

Finally, the set $\mathcal{CFG}$ contains all valid pairings of articulators and configurations, such that $\mathcal{CFG} = \{(a, c) : a \in \mathcal{ART} \text{ and } c \in \mathcal{CFG}^a\}$. ■

Location and configuration traits alone are not enough to characterize SL utterances; in order to give a more complete depiction of language, concepts linked to spatial

relationships such as position, movement and distance have to be taken in account. This will be done mainly by way of direction vectors, which are defined below.

Definition 4.6 (Set of directions)**.** Let $\Delta \subseteq \Psi_{\mathtt{SL}} \times \Psi_{\mathtt{SL}}$ be the set containing every pair of coordinates in $\Psi_{\mathtt{SL}}$; for each pair $(A, B) \in \Delta$, where $A, B \in \Psi_{\mathtt{SL}}$, the symbol $\overrightarrow{AB}$ will denote the vector going from coordinate A to B. ■

By itself, the set Δ holds no particular meaning whatsoever, as it is for the annotator to determine the relevance of directions himself. In general, this can be done by classifying vectors under more meaningful classes such like "`LEFT`" or "`RIGHT`". To this end, it is useful to introduce a labeling function taking groups of vectors to names, which will serve to characterize spatial concepts over the coordinate system induced by $\Psi_{\mathtt{SL}}$.

Definition 4.7 (Labeling function for Δ)**.** Let $\mathcal{DIRECTIONS} \subset \mathcal{NAMES}$ be the set of direction names, such that $\mathcal{DIRECTIONS} \cap \mathcal{BODY} = \emptyset$ and, for any $a \in \mathcal{ART}$, $\mathcal{DIRECTIONS} \cap \mathcal{CFG}^a = \emptyset$. The mapping $\mathcal{L}_\Delta : 2^\Delta \to \mathcal{DIRECTIONS}$ denotes the labeling function taking groups of vectors in Δ to a name in $\mathcal{DIRECTIONS}$; that is, $\mathcal{L}_\Delta$ assigns a single name to a set of vectors defining a direction. ■

The labeling function of Definition 4.7 can be used to populate yet another set, containing information of linguistic interest for sets of vectors.

Definition 4.8 (Set of relevant directions for SL)**.** The set of relevant directions for SL is defined as follows:

$$\Delta_{\mathtt{SL}} = \{v : v \in 2^\Delta \text{ and there is a name } t \in \mathcal{L}_\Delta(v)\}$$

Notably, $v \in \Delta_{\mathtt{SL}}$ denotes a vector family identified by a name in $\mathcal{DIRECTIONS}$. ■

Intuitively, the set in Definition 4.8 contains only the vector families defined over $\mathcal{L}_\Delta$, the labeling function from Definition 4.7; in this way, if definitions for $\mathtt{LEFT}, \mathtt{RIGHT} \in \mathcal{DIRECTIONS}$ exist in $\mathcal{L}_\Delta$, the group of vectors giving meaning to these names will be in $\Delta_{\mathtt{SL}}$. This enables annotators to separate geometric information from spatial concepts, thus permitting him to work with words like `LEFT` and `RIGHT` instead of defining groups of vectors.

These initial definitions are but the building blocks of representation; they are intended to be combined with each other, in order to describe more complex relationships. This combination is achieved by way of the $\mathrm{PDL}_{\mathtt{SL}}$ grammar, which is presented in the next section.

4.2 Syntax

The previously defined primitives, serve the purpose of providing some semantic information about posture parameters before characterizing postures themselves; to properly represent these, it is necessary to give *statements*, declare information about them that can be proven true or false. Roughly, the idea is that each statement will represent individual information pieces about articulators, and postures will be none other than the combination of several of these pieces. As such, it is useful to formally define these statements as atomic propositions, like shown in Definition 4.9.

Definition 4.9 (Atomic Propositions for SL Body Articulators)**.** Let $a \in \mathcal{ART}$ be an articulator. The set Φ^a, containing the *atomic propositions for articulator* a, is defined as follows:

$$\Phi^a = \{p^a : p^a \text{ asserts information about the articulator named by } a \in \mathcal{ART}\}$$

Also, for every articulator $a \in \mathcal{ART}$, there exists a proposition $\mathbf{any}^a \in \Phi^a$ denoting that *articulator* a *doesn't move.*

The set of all possible atomic propositions is defined as follows:

$$\Phi = \bigcup_{a \in \mathcal{ART}} \Phi^a$$

■

The propositions defined in set Φ, are intended to characterize phonetic parameters; that is, they strive to assert propositional information about Holds in the Movement-Hold Model (MHM) phonological model (see Chapter 2, Subsection 2.2.1.4, 29). In a similar fashion, Movements will be characterized independently, by a set denoting actions.

Definition 4.10 (Atomic Actions for SL Body Articulators)**.** Let $a \in \mathcal{ART}$ be an articulator. The set Π^a of *atomic actions for articulator* a, is defined as:

$$\Pi^a = \{\pi^a : \pi^a \text{ asserts a change in articulator } a \in \mathcal{ART}\}$$

In addition, for every articulator $a \in \mathcal{ART}$, there exists an atomic action $\mathbf{skip}^a \in \Pi^a$ denoting that *articulator* a *performs any action.*

Finally, the set of all possible atomic actions is defined as follows:

$$\Pi = \bigcup_{a \in \mathcal{ART}} \Pi^a$$

■

It is important to underline that members of both sets, Φ and Π, are statements; that is, each member can be labeled as true or false, as specified by PDL_{SL} semantics (see Section 4.3). This makes it possible to combine multiple statements into more complex phonological structures, starting with actions.

Definition 4.11 (Action Language for SL Body Articulators $\mathcal{A}_{\mathsf{SL}}$)**.** The *action language for SL body articulators* ($\mathcal{A}_{\mathsf{SL}}$) is given by the following rule:

$$\alpha ::= \pi^a \mid \alpha \cap \alpha \mid \alpha \cup \alpha \mid \alpha; \alpha$$

where $\pi^a \in \Pi$.

Intuitively:

— π^a is an atomic action for some articulator $a \in \mathcal{ART}$;
— $\alpha \cap \alpha$ indicates the concurrent execution of two actions;
— $\alpha \cup \alpha$ is used to depict two non-deterministically occurring actions;
— $\alpha; \alpha$ describes a sequence of two actions; and,

■

Definition 4.11, presents the PDL_{SL} action language as a Backus Normal Form (BNF) grammar; the operations depicted, intend to give the annotator the flexibility to combine various atomic actions into a single Movement, to describe complex phonological changes. For instance, it permits describing situations where two hands move simultaneously, where a signer repeats an action an unbounded sequence of times or even if there might be more than one valid changes in the same interval. The resulting formulae can further be "glued" with Hold representations by way of the PDL_{SL} language, defined below.

Definition 4.12 (Language PDL_{SL})**.** The language PDL_{SL} is generated by the following rule:

$$\varphi ::= \top \mid p^a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\alpha\rangle\varphi \mid \langle\alpha \wedge \varphi\rangle\varphi \mid \varphi; \langle\alpha\rangle\varphi$$

where $p^a \in \Phi$, $\alpha \in \mathcal{A}_{\mathsf{SL}}$.

Intuitively:

— $\top$ represents a logical true value;
— p^a is an atomic proposition for articulator $a \in \mathcal{ART}$;
— $\neg\varphi$ represents the negation of formula φ;

— $\varphi \wedge \varphi$ denotes conjunction;
— $\langle\alpha\rangle\varphi$ describes that, after execution of action α, φ holds;
— $\langle\alpha \wedge \varphi\rangle\varphi$ models situations where an action α and an static feature φ occur simultaneously until converging into some other static feature; and,
— $\varphi;\langle\alpha\rangle\varphi$ denotes that, if some φ holds, then a modal formula holds afterwards.

■

Just as $\mathcal{A}_{\mathsf{SL}}$ enables the construction of more complex Movements, the language PDL_{SL} permits the integration of groups of atomic propositions into Holds. In addition, it provides operations like $\langle\alpha\rangle\varphi$ to think in terms of sequences of Movements and Holds, giving way to the description of phonological structures. In this regard, PDL_{SL} differs from PDL on the interpretation of modal operators, as PDL_{SL} is essentially constrained by the possible phonetic sequences occurring in the studied SL. As a result, PDL_{SL} formulae will be interpreted over a mathematical representation of SL utterances, designed to capture the effects of multimodal articulation; the interpretation rules, and the proposed utterance representation, are introduced in the next section.

4.3 Semantics

In the spirit of the possible worlds model introduced by [Hintikka 1962], formulae in the PDL_{SL} language are interpreted mostly over Labeled Transition Systems (LTSs); however, concurrent operators will hold meaning only over more complex structures. In the case of SLs, this representation will reflect the possible morphological states signers can assume, depending on how they configure their articulation parameters during signing. But first, to know which state changes are valid in a determined SL, it is necessary to define an *articulation model*, corresponding to a LTS of Holds connected by possible Movements with their outcomes. The formal definition is presented below.

Definition 4.13 (Sign Language Articulation Model)**.** Let $a \in \mathcal{ART}$ be an articulator, the *articulation model of* a ($\mathcal{M}^a$) is a tuple $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ where:

— S is a non-empty set of states
— R is a transition relation $R \subseteq S \times S$ where, for every $s \in S$, there exists a state $s' \in S$ such that $(s, s') \in R$. Furthermore, it can be the case that $s = s'$.
— $[\![\cdot]\!]_{\Phi^a} : S \to 2^{\Phi^a}$, denotes the function mapping states to a set of atomic propositions giving information about articulator a. Notably, for states $s, s' \in S$, if $[\![s]\!]_{\Phi^a} \equiv [\![s']\!]_{\Phi^a}$, then $s \equiv s'$.
— $[\![\cdot]\!]_{\Pi^a} : R \to 2^{\Pi^a}$, denotes the function mapping binary relations to groups of atomic actions for articulator a.

■

It is implicit from the definition that each articulator has its own model, in order to reflect changes in individual articulators rather than on entire postures. Parametric information is provided by the labeling functions, $[\![\cdot]\!]_{\Phi^a}$ and $[\![\cdot]\!]_{\Pi^a}$, which are designed to restrain which information states and transitions may hold, respectively. It is by way of these labellings that the data structure gains meaning, as they serve to relate linguistic parameters to the model: states are labeled with information about Holds, while transitions are tagged with data pertaining to Movements. Thus, a path in the articulation model of $a \in \mathcal{ART}$ can be interpreted as a sequence of Holds and Movements containing only information about a. A formal characterization of such a path, is given by Definition 4.14.

Definition 4.14 (Articulation Sequence)**.** Let $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ be an articulation model for articulator $a \in \mathcal{ART}$. An *articulation sequence of* $\mathcal{M}^a$, is a finite ordered list of transitions $\mathcal{S}^a = ((s_0, s_1), (s_1, s_2), \ldots, (s_n, s_{n+1}))$, such that:

— if $(s_k, s_{k+1}) \in \mathcal{S}^a$, then $(s_k, s_{k+1}) \in R$;

— if transitions (s_i, s_{i+1}) and (s_j, s_{j+1}) are adjacent in $\mathcal{S}^a$, then $[\![s_{i+1}]\!]_{\Phi^a} \equiv [\![s_j]\!]_{\Phi^a}$.

It is said that $\mathcal{M}^a$ *induces* sequence $\mathcal{S}^a$, or that the latter *is induced* by the former. ■

Roughly, $\mathcal{S}^a$ describes a phoneme sequence where only a participates, expressed by way of state changes over a LTS. Even though time duration is not expressed by the sequence, there is a correspondence between phonemes as time-intervals and $\mathcal{S}^a$, as depicted on Figure 4.2.

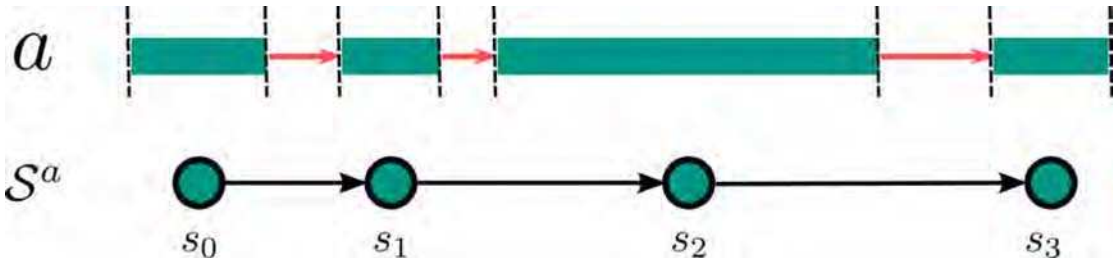


FIGURE 4.2: Representation of articulation sequence $\mathcal{S}^a$ for changes in articulator $a \in \mathcal{ART}$.

The upper part of the diagram in Figure 4.2 corresponds to a chain of time intervals, depicted as an interleaving of rectangles and arrows. In the figure, each rectangle represents a Hold, while each arrow corresponds to a Movement. Below the interval chain, the sequence $\mathcal{S}^a$ depicts the same information under a different representation; here, each Movement is shown as two states connected by an arrow, implicitly equating Holds to states. However, it is important to note that articulation sequences will most of

the time represent partial phonemes, as they only take in account changes of a single articulator.

Although Hold and Movement intervals have an implicit duration, articulation sequences provide no way to account for time. This situation is addressed by Definition 4.15, where a time-stamp operator is introduced to articulation sequences.

Definition 4.15 (Timed articulation sequence)**.** Let $\mathcal{S}^a$ be an articulation sequence of $a \in \mathcal{ART}$, and let $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ be the articulation model inducing $\mathcal{S}^a$. A *timed articulation sequence of* a, is a tuple $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a)$, where $\mathcal{T}^a : S \cup R \to \mathbb{N} \times \mathbb{N}$ is a labeling function, mapping model states and transitions appearing in $\mathcal{S}^a$ to time intervals such that:

— for any pair of adjacent transitions $(s_{k-1}, s_k), (s_k, s_{k+1}) \in \mathcal{S}^a$, with allotted time-intervals $\mathcal{T}^a((s_{k-1}, s_k)) = [t_i, t_j)$ and $\mathcal{T}^a((s_k, s_{k+1})) = [t_m, t_n)$, it holds that $1 < t_i < t_j$, $t_m < t_n$ and $t_j < t_m$;
— for each state s_k common to each pair of adjacent transitions, the state's allotted time duration is $\mathcal{T}^a(s_k) = [t_j, t_m)$. In addition, if $(s_{k-1}, s_k) \in \hat{\mathcal{S}}^a$ is the first transition in the sequence, then state s_{k-1} has a duration of $\mathcal{T}^a(s_{k-1}) = [1, t_i)$.

■

For simplicity, in the rest of this document $e_i \in \hat{\mathcal{S}}^a$ will denote element e_i from articulation sequence $\mathcal{S}^a \in \hat{\mathcal{S}}^a$, unless indicated otherwise.

It is brought into attention that, for the rest of this thesis, intervals are defined in terms of natural numbers (*i.e.* positive integers). This is, interval $I = [m, n)$ is to be interpreted as the set $\{m, m+1, m+2, \ldots, n-1\}$, $m, n \in \mathbb{N}$ and $m < n$ where the square bracket next to the m indicates that $m \in I$, while the parenthesis next to the n indicates that $n \notin I$. Similarly, two intervals overlap if the sets have numbers in common. For example, intervals $[m, k)$ and $[k, n)$ don't overlap, whereas intervals $[1, k]$ and $[k, n)$ do (they have k in common). Finally, interval $[m, m]$ is defined, whereas $[m, m)$ and $(m, m]$ are not.

Intuitively, timed articulation sequences induce a chain of strictly ordered intervals, which is proven below.

Lemma 4.1. *Let $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a)$ be a timed articulation sequence; $\mathcal{T}^a$ induces a strict interval order over states and transitions represented by $\mathcal{S}^a$.*

Proof. By induction. Let $(s_0, s_1) \in \mathcal{S}^a$ be the first transition in the sequence. By Definition 4.15, $\mathcal{T}^a$ labels state s_0 and transition (s_0, s_1) as $\mathcal{T}^a(s_0) = [1, t)$ and $\mathcal{T}^a((s_0, s_1)) = [t, t')$, respectively. Therefore, it holds for $\mathcal{T}^a(s_0) < \mathcal{T}^a((s_0, s_1))$. Assume it holds for

the interval sequence $\mathcal{T}^a(s_0) < \mathcal{T}^a((s_0, s_1)) < \mathcal{T}^a(s_1) < \ldots < \mathcal{T}^a((s_{k-1}, s_k)) < \mathcal{T}^a(s_k)$. For $\mathcal{T}^a(s_k, s_{k+1})$, we know by Definition 4.15 that the state s_k, common to two adjacent transitions, is labeled as follows: if $\mathcal{T}^a(s_{k-1}, s_k) = [t_i, t_j)$ and $\mathcal{T}^a(s_k, s_{k+1}) = [t_m, t_n)$, then $\mathcal{T}^a(s_k) = [t_j, t_m)$. Thus, interval $\mathcal{T}^a(s_k, s_{k+1})$ necessarily starts at time t_m, where $\mathcal{T}^a(s_k) < t_m$, which implies that $\mathcal{T}^a(s_k) < \mathcal{T}^a(s_k, s_{k+1})$. Then, for $\mathcal{T}^a(s_k, s_{k+1})$, it holds. □

In some cases, the parameters given by a single timed articulation sequence are enough to model complete phonological sequences; for instance, when representing one handed signs, the information of a single hand may be all that's needed to characterize a sign. For this reason, it is useful to give semantics of PDL_{SL} formulae in terms of timed articulation sequences, as shown in Definition 4.16.

Definition 4.16 (PDL_{SL} formulae satisfaction over timed articulation sequences)**.** Let $\hat{\mathcal{S}}^a$ be a timed articulation sequence and let $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ be the articulation model inducing $\mathcal{S}^a \in \hat{\mathcal{S}}^a$. Also, let φ and ψ be PDL_{SL} formulae and $p^a \in \Phi^a$. The semantics of the non-modal PDL_{SL} operators on Definition 4.12, are given as follows:

$$
\begin{array}{lll}
\mathcal{M}^a, s_k \models \top & & \text{always holds;} \\
\mathcal{M}^a, s_k \models p^a & \textit{iff} & p^a \in [\![s_k]\!]_{\Phi^a}; \\
\mathcal{M}^a, s_k \models \neg\varphi & \textit{iff} & \mathcal{M}^a, s_k \not\models \varphi; \\
\mathcal{M}^a, s_k \models \varphi \wedge \psi & \textit{iff} & \mathcal{M}^a, s_k \models \varphi \text{ and } \mathcal{M}^a, s_k \models \psi.
\end{array}
$$

Let $\overline{e_{ij}} = e_i, e_{i+1}, \ldots, e_j \in \hat{\mathcal{S}}^a$, $i \leq j$ be a chain of adjacent elements in the sequence $\hat{\mathcal{S}}^a$, and let s_k denote any state in any transition $e_i \in \hat{\mathcal{S}}^a$. Also, note that $\pi^a \in \Pi^a$ and $\alpha, \beta \in \mathcal{A}_{\mathsf{SL}}$. Satisfaction over timed sequences is given as follows:

$$
\begin{array}{lll}
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models \top & & i = j,\ e_i = (s_k, s_{k+1}), \text{ and } \mathcal{M}^a, s_k \models \top; \\
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models p^a & \textit{iff} & i = j,\ e_i = (s_k, s_{k+1}), \text{ and } \mathcal{M}^a, s_k \models p^a; \\
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models \neg\varphi & \textit{iff} & i = j,\ e_i = (s_k, s_{k+1}), \text{ and } \mathcal{M}^a, s_k \models \neg\varphi; \\
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models \varphi \wedge \psi & \textit{iff} & i = j,\ e_i = (s_k, s_{k+1}), \text{ and } \mathcal{M}^a, s_k \models \varphi \wedge \psi; \\
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models \langle\pi^a\rangle\varphi & \textit{iff} & i = j,\ e_i = (s_k, s_{k+1}),\ \pi^a \in [\![(s_k, s_{k+1})]\!]_{\Pi^a} \\
 & & \text{and } \mathcal{M}^a, s_{k+1} \models \varphi; \\
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models \langle\alpha;\beta\rangle\varphi & \textit{iff} & \hat{\mathcal{S}}^a, \overline{e_{ij}} \models \langle\alpha\rangle\langle\beta\rangle\varphi; \\
\hat{\mathcal{S}}^a, \overline{e_{ij}} \models \psi;\langle\alpha\rangle\varphi & \textit{iff} & \hat{\mathcal{S}}^a, \overline{e_{ik}} \models \psi \text{ and } \hat{\mathcal{S}}^a, \overline{e_{kj}} \models \langle\alpha\rangle\varphi, \text{ where} \\
 & & i \leq k \leq j.
\end{array}
$$

■

The missing PDL_{SL} operators cannot be defined in terms of timed articulation sequences, as their meaning is related to the interaction between two or more articulators, whereas each $\hat{\mathcal{S}}^a$ holds information about a single articulator. To address this situation, Definition 4.17 introduces a new data structure built upon sets of timed articulation sequences, enabling the synchronization of multiple channels into phoneme chains.

Definition 4.17 (SL Utterance)**.** A *SL utterance* is a tuple $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ where:

— $\mathcal{P}$ is an arbitrarily long list of ordered phonemes $p_0, p_1, \ldots, p_n$;
— $\mathcal{T}_\mathcal{U} : \mathcal{P} \to \mathbb{N} \times \mathbb{N}$ is a time-labeling function for phonemes;
— $[\![\cdot]\!]_\mathcal{U} : \mathcal{P} \to 2^{\Phi \cup \Pi}$ labels phonemes to a set of atomic propositions and actions (a more thorough description is given in Definition 4.19);
— $\mathcal{SEQ}$ is a group of timed articulation sequences such that, for each $a \in \mathcal{ART}$, there is exactly one sequence $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a) \in \mathcal{SEQ}$.

The list $\mathcal{P}$ is time-labeled recursively as follows:

1. For p_0, let $e_0, e_1, \ldots, e_n = \hat{\mathcal{S}}^\circ \in \mathcal{SEQ}$ be the timed articulation sequence holding the earliest occurring transition of all; this is, $\mathcal{T}^\circ(e_0) = [t, t')$ has the smallest t of every sequence in $\mathcal{SEQ}$. The phoneme p_0 represents the first occurring phoneme in the utterance, and is time-labeled as $\mathcal{T}_\mathcal{U}(p_0) = [1, t)$, $1 < t$.
2. For any other p_N, $N > 0$, such that $\mathcal{T}_\mathcal{U}(p_{N-1}) = [t_i, t_j)$, p_N is time-labeled as $\mathcal{T}_\mathcal{U}(p_N) = [t_j, t_k)$, where t_k is either the first instant after some transition ends, or the first instance where a new transition starts, and $t_i < t_j < t_k$.

■

In general, utterances synchronize information with respect to global articulatory changes; intuitively, utterances strive to maintain a strict ordering across all sequences in $\mathcal{SEQ}$, dividing each sequence if necessary when concurrent articulations show changes. In this sense, phonemes can be seen as "cuts" of time where articulation parameters remain essentially static. Figure 4.3, shows the visual representation of a utterance $\mathcal{U}$, for three articulators.

In the figure, the bottom part shows the chain of states representing utterance phonemes, aligned with the interval segments to which they correspond. Just above, a representation of the individual sequences contained in $\mathcal{SEQ}$ is given for three articulators $a_1, a_2, a_3 \in \mathcal{ART}$; furthermore, each state and transition represented in $\mathcal{SEQ}$ has marked the phonemes in $\mathcal{P}$ to which they correspond.

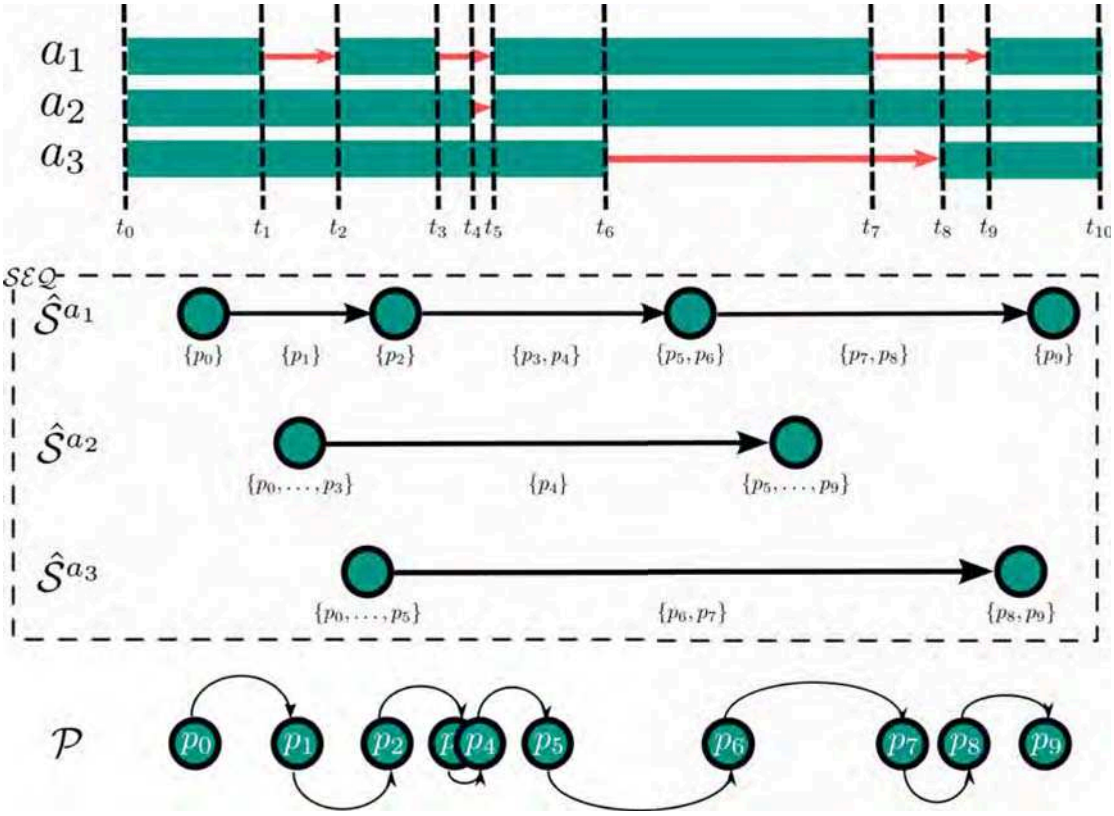


FIGURE 4.3: Representation of a case where a utterance object synchronizes three articulation sequences.

Utterances make possible the combination of propositional information from different channels, regardless of whether it corresponds to static or change parameters. For this reason, it is important to ensure that articulators will retain consistent properties when the information is combined into phonemes; notably, it is important to be certain that a phoneme will not hold contradicting information about each articulator. In this respect, utterances guarantee that articulation changes won't occur inside phonemes, but rather than they will be generated around them, as shown by the following lemmas.

Lemma 4.2. *Let $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ be an utterance. For each phoneme $p_N \in \mathcal{P}$, and any sequence $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a) \in \mathcal{SEQ}$, utterance interval $\mathcal{T}_\mathcal{U}(p_N)$ won't overlap with two adjacent intervals in the chain induced by $\mathcal{T}^a$.*

Proof. By contradiction. Suppose that $\mathcal{T}_\mathcal{U}(p_N) = [t_i, t_j)$ overlaps with two adjacent intervals $\mathcal{T}^a(s_k)$ and $\mathcal{T}^a(s_{k+1})$. By Lemma 4.1, it is safe to assume that $\mathcal{T}^a(s_k) = [t_x, t_y)$ and $\mathcal{T}^a((s_k, s_{k+1})) = [t_y, t_z)$. An overlapping exists if and only if $t_i \in [t_x, t_y)$ and there exist some $t' \in [t_i, t_j)$, such that $t_i < t_y \leq t' \leq t_{z-1}$. Note that t_y is the starting time of a new transition in $\mathcal{SEQ}$; by Definition 4.17, interval $\mathcal{T}_\mathcal{U}(p_N)$ can be only as long as $\mathcal{T}_\mathcal{U}(p_N) = [t_i, t_y)$, which occurs when t_y is the first transition change after t_i. Implicitly,

this means that every t' will be delimited by $t_i < t' < t_y$, thus implying that $\mathcal{T}_\mathcal{U}(p_N)$ doesn't overlap with $\mathcal{T}^a((s_k, s_{k+1})) = [t_y, t_z)$! □

Lemma 4.3. *Let $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ be an utterance. For each phoneme $p_N \in \mathcal{P}$, and any sequence $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a) \in \mathcal{SEQ}$, utterance interval $\mathcal{T}_\mathcal{U}(p_N)$ won't overlap with two non-adjacent intervals in the chain induced by $\mathcal{T}^a$.*

Proof. By contradiction, suppose that $\mathcal{T}_\mathcal{U}(p_N) = [t_i, t_j)$ overlaps with two non-adjacent intervals $\mathcal{T}^a((s_{k-1}, s_k))$ and $\mathcal{T}^a(s_k, s_{k+1})$, separated by interval $\mathcal{T}^a(s_k)$. Note that

$$\mathcal{T}^a((s_{k-1}, s_k)) < \mathcal{T}^a(s_k) < \mathcal{T}^a((s_k, s_{k+1})),$$

with respective values $\mathcal{T}^a((s_{k-1}, s_k)) = [t_v, t_w)$, $\mathcal{T}^a(s_k) = [t_w, t_x)$ and $\mathcal{T}^a((s_k, s_{k+1})) = [t_x, t_y)$. An overlapping exists if and only if $t_i \in [t_v, t_w)$ and there is some $t' \in [t_i, t_j)$, such that $t_i < t_x \leq t' \leq t_{y-1}$. However, by Lemma 4.2 we know that no such t' exists as it would imply that $t_i < t_w < t' \leq t_{j-1}$! Thus, $\mathcal{T}_\mathcal{U}(p_N)$ doesn't overlap with $\mathcal{T}^a(s_k, s_{k+1})$.

By the same reasoning, it follows that $\mathcal{T}_\mathcal{U}(p_N)$ won't overlap with any farther non-adjacent interval; let $\mathcal{T}^a((s_n, s_{n+1}))$ be any interval such that

$$\mathcal{T}^a((s_{k-1}, s_k)) < \mathcal{T}^a(s_k) < \mathcal{T}^a((s_k, s_{k+1})) < \ldots < \mathcal{T}^a((s_n, s_{n+1})),$$

Lemma 4.2 implies that $t' < t_w$ and, since $\hat{\mathcal{S}}^a$ is strictly ordered, it is not possible that $\mathcal{T}_\mathcal{U}(p_N)$ and $\mathcal{T}^a((s_n, s_{n+1}))$ overlap. □

Corollary 4.1. *For utterance $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$, each phoneme $p_N \in \mathcal{P}$ overlaps with a single interval of the chain induced by $\mathcal{T}^a$, where $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a) \in \mathcal{SEQ}$.* ■

Once proven that information won't change inside phonemes, it is safe to define phonetic parameters in terms of utterances; as presented in Definition 4.17, phonemes are labeled by way of a mapping $[\![\cdot]\!]_\mathcal{U}$, the details of which are explained by the next two Definitions.

Definition 4.18 (Phoneme to sequence mapping)**.** Let $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ be an utterance and $p_N \in \mathcal{P}$ be a phoneme labeled with time interval $\mathcal{T}_\mathcal{U}(p_N)$. Let $\hat{\mathcal{S}}^a \in \mathcal{SEQ}$ be the timed articulation sequence in $\mathcal{U}$ corresponding to articulator $a \in \mathcal{ART}$, while $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ denotes the articulation model inducing $\hat{\mathcal{S}}^a$. Also, let $\mathcal{T}^a$ be the time-labeling function associated to $\hat{\mathcal{S}}^a$. Function $\mathcal{V}^a : \mathcal{P} \to S \cup R$, mapping phonemes to transitions or states of the articulation model $\mathcal{M}^a$, is defined as follows:

$$\mathcal{V}^a(p_N) = \begin{cases} e_k & \text{if } \mathcal{T}_\mathcal{U}(p_N) \text{ overlaps with } \mathcal{T}^a(e_k), \text{ where } e_k \in \hat{\mathcal{S}}^a; \\ s_k & \text{if } \mathcal{T}_\mathcal{U}(p_N) \text{ overlaps with } \mathcal{T}^a(s_k), \text{ where } s_k \text{ is a state common} \\ & \text{to two adjacent transitions } (s_{k-1}, s_k), (s_k, s_{k+1}) \in \hat{\mathcal{S}}^a. \end{cases}$$

Note that Corollary 4.1 guarantees that $\mathcal{T}_\mathcal{U}(p_N)$ overlaps with either a single transition, or a single state. ■

Definition 4.19 (Utterance phonetic parameters)**.** Let $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ be an utterance and $p_N \in \mathcal{P}$ be a phoneme labeled with time interval $\mathcal{T}_\mathcal{U}(p_N)$. Let $\hat{\mathcal{S}}^a \in \mathcal{SEQ}$ be the timed articulation sequence in $\mathcal{U}$ corresponding to articulator $a \in \mathcal{ART}$, while $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ denotes the articulation model inducing $\hat{\mathcal{S}}^a$. The mapping $[\![\cdot]\!]^a_\mathcal{U} : \mathcal{P} \to 2^{\Phi^a \cup \Pi^a}$, denoting the parametric information of articulator $a \in \mathcal{ART}$ at interval $\mathcal{T}_\mathcal{U}(p_N)$, is defined as follows:

$$[\![p_N]\!]^a_\mathcal{U} = \begin{cases} [\![\mathcal{V}^a(p_N)]\!]_{\Pi^a} & \text{if } \mathcal{V}^a(p_N) \in R; \\ [\![\mathcal{V}^a(p_N)]\!]_{\Phi^a} & \text{if } \mathcal{V}^a(p_N) \in S; \end{cases}$$

where $\mathcal{V}^a$ is the mapping given on Definition 4.18.

The function giving the set of *phonetic parameters* of $p_N \in \mathcal{P}$ is then defined as:

$$[\![p_N]\!]_\mathcal{U} = \bigcup_{a \in \mathcal{ART}} [\![p_N]\!]^a_\mathcal{U}$$

■

Finally, Theorem 4.4 shows that utterances take in account every articulatory change; this is, that if some transition occurs between two parametric states, then it will be contained in an utterance's phoneme.

Theorem 4.4 (Utterance phoneme to transition correspondence)**.** *Let $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ be an utterance and $\hat{\mathcal{S}}^a$ denote a timed articulation sequence of $a \in \mathcal{ART}$. For every $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a) \in \mathcal{SEQ}$ and for each transition $(s_k, s_{k+1}) \in \hat{\mathcal{S}}^a$, there exist a $p_N \in \mathcal{P}$ in the phoneme sequence such that $[\![(s_k, s_{k+1})]\!]_{\Pi^a} \subseteq [\![p_N]\!]_\mathcal{U}$.*

Proof. By contradiction. Suppose that there exist a timed articulation sequence $\hat{\mathcal{S}}^a$ where some transition $(s_k, s_{k+1}) \in \hat{\mathcal{S}}^a$ is not represented in $\mathcal{P}$. If this is the case, we know (s_k, s_{k+1}) was not the first transition in $\mathcal{SEQ}$, or it would have been added as per Definition 4.17.

Since (s_k, s_{k+1}), time-stamped as $\mathcal{T}^a((s_k, s_{k+1})) = [t_i, t_j)$, is not represented in the sequence, it means that there is a phoneme p_N which was time-stamped as $\mathcal{T}_\mathcal{U}(p_N) = [t_h, t_i)$ and a phoneme p_{N+1} which was time-stamped as $\mathcal{T}_\mathcal{U}(p_{N+1}) = [t_j, t_k)$. However, if $\mathcal{T}_\mathcal{U}(p_N) = [t_h, t_i)$ then, by the definition of $\mathcal{U}$ it is the case that $\mathcal{T}_\mathcal{U}(p_{N+1})$ must have started at time t_i, because we know that t_i it is the beginning of transition (s_k, s_{k+1})! Also, note that if phoneme p_{N+1} starts at time t_i, it will either end at time t_{j-1} or before. Thus $[\![(s_k, s_{k+1})]\!]_{\Phi^a} \subseteq [\![p_{N+1}]\!]_\mathcal{U}$. □

Corollary 4.2. *For every $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a) \in \mathcal{SEQ}$ and for each transition $(s_k, s_{k+1}) \in \hat{\mathcal{S}}^a$, there exist a $p_N \in \mathcal{P}$ in the phoneme sequence such that $[\![s_{k+1}]\!]_{\Phi^a} \subseteq [\![p_N]\!]_\mathcal{U}$.* ■

Corollary 4.3. *Phonemes in $\mathcal{U}$ induce a total order over transitions in $\mathcal{SEQ}$.* ■

With these results, it becomes apparent that there will always be a *consistent* phoneme in $\mathcal{U}$ for transition in $\mathcal{SEQ}$; that is, phonemes will neither omit information nor contain contradicting propositions, as long as articulation sequences themselves remain consistent. Finally, Lemma 4.5 provides an idea of how many phonemes are needed to build $\mathcal{U}$.

Lemma 4.5 (Number of states in $\mathcal{U}$)**.** *Let n denote the total number of transitions in all sequences of $\mathcal{SEQ}$. Utterance $\mathcal{U}$ assigns, at most, $2n + 1$ phonemes to represent all the information in $\mathcal{SEQ}$.*

Proof. Theorem 4.4 implies that, if transitions don't overlap, each of them will induce the time-stamping of, at least, one phoneme. Recall by Definition 4.17 that phonemes, except from the first one, will always be time-stamped either when a transition's interval begins or ends. In addition, from the definition, if the start of a transition's interval induces the time-stamping of a phoneme, then said phoneme will have, at most, the same length than the interval, at which moment a new phoneme will be issued. This implies that every non-overlapping transition will necessarily induce the time-stamping 2 phonemes. Also, note that if k transitions overlap, $k \geq 2$, then it can be the case that:

- they perfectly overlap, in which case a single phoneme contains the information of all the transitions;
- they don't perfectly overlap, in which case at most $2k$ phonemes will be created, if none of their starting or ending points are aligned.

Finally, note by Corollary 4.3 that transitions are totally ordered, implying that they will only be tagged once by $\mathcal{U}$. Thus, in the worst case, $\mathcal{U}$ will time-stamp $2n$ phonemes, two for each transition, plus the phoneme in the beginning, giving $2n + 1$. □

Overall, these properties permit defining the semantics of concurrent PDL$_{\mathsf{SL}}$ formulae over utterances, in terms of phonological change; the proposed satisfaction rules are presented in Definition 4.20.

Definition 4.20 (PDL_{SL} formulae satisfaction over utterances)**.** Let $\mathcal{U} = (\mathcal{P}, \mathcal{T}_{\mathcal{U}}, [\![\cdot]\!]_{\mathcal{U}}, \mathcal{SEQ})$ be an utterance and $p_N \in \mathcal{P}$ be a phoneme labeled with time interval $\mathcal{T}_{\mathcal{U}}(p_N)$. Let $\hat{\mathcal{S}}^a \in \mathcal{SEQ}$ be the timed articulation sequence in $\mathcal{U}$ corresponding to articulator $a \in \mathcal{ART}$, while $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ denotes the articulation model inducing $\hat{\mathcal{S}}^a$. Also, let φ and ψ be PDL_{SL} formulae and $p^a \in \Phi^a$.

Let $\overline{p_{ij}} = p_i, p_{i+1}, \ldots, p_j \in \mathcal{P}$, $i \leq j$ be a segment of phoneme list $\mathcal{P}$ starting in p_i. Similarly, let $\overline{e_{ij}} = e_i, e_{i+1}, \ldots, e_j \in \hat{\mathcal{S}}^a$, $i \leq j$ be a chain of adjacent transitions in the sequence $\hat{\mathcal{S}}^a$. For simplicity, if $i = j$ then $\overline{p_{ij}} \equiv \overline{p_i}$ and $\overline{e_{ij}} \equiv \overline{e_i}$. Satisfaction over utterance phonemes is given as follows:

$\mathcal{U}, \overline{p_{ij}} \models$	$\top$	*iff*	$i = j$, always holds;
$\mathcal{U}, \overline{p_{ij}} \models$	p^a	*iff*	$i = j$, $s_k = \mathcal{V}^a(p_i) \in S$ where $e_m = (s_k, s_{k+1}) \in \hat{\mathcal{S}}^a$ is the transition associated to s_k in $\hat{\mathcal{S}}^a$ and $\hat{\mathcal{S}}^a, \overline{e_m} \models p^a$ holds;
$\mathcal{U}, \overline{p_{ij}} \models$	$\neg\varphi$	*iff*	$i = j$, and $\mathcal{U}, \overline{p_i} \not\models \varphi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\varphi \wedge \psi$	*iff*	$i = j$, where $\mathcal{U}, \overline{p_i} \models \varphi$ and $\mathcal{U}, \overline{p_i} \models \psi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\langle \pi^a \rangle \varphi$	*iff*	$e_m = \mathcal{V}^a(p_{i+1}) \in \hat{\mathcal{S}}^a$ where $e_m = (s_k, s_{k+1})$ such that $\hat{\mathcal{S}}^a, \overline{e_m} \models \langle \pi^a \rangle \top$, and for $\mathcal{V}^a(p_N) \equiv s_{k+1}$, where $i < N \leq j$, it holds $\mathcal{U}, \overline{p_{Nj}} \models \varphi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\langle \alpha; \beta \rangle \varphi$	*iff*	$\mathcal{U}, \overline{p_{ij}} \models \langle \alpha \rangle \langle \beta \rangle \varphi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\langle \alpha \cup \beta \rangle \varphi$	*iff*	$\mathcal{U}, \overline{p_{ij}} \models \langle \alpha \rangle \varphi$ or $\mathcal{U}, \overline{p_{ij}} \models \langle \beta \rangle \varphi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\langle \alpha \cap \beta \rangle \varphi$	*iff*	$\mathcal{U}, \overline{p_{iN}} \models \langle \alpha \rangle \top$, where $i < N \leq j$ and $\mathcal{U}, \overline{p_{iM}} \models \langle \beta \rangle \top$, where $i < M \leq j$ such that, — if $N < M$ then $\mathcal{U}, \overline{p_{N+1M}} \models \neg\langle \alpha \rangle \top$ and $\mathcal{U}, \overline{p_{Mj}} \models \varphi$; — if $M < N$ then $\mathcal{U}, \overline{p_{M+1N}} \models \neg\langle \beta \rangle \top$ and $\mathcal{U}, \overline{p_{Nj}} \models \varphi$; — if $M = N$ then $\mathcal{U}, \overline{p_{Nj}} \models \varphi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\langle \alpha \wedge \psi \rangle \varphi$	*iff*	$\mathcal{U}, \overline{p_{iN}} \models \langle \alpha \rangle \top$, where $i < N \leq j$ such that $\mathcal{U}, \overline{p_{iN-1}} \models \psi$, and $\mathcal{U}, \overline{p_{Nj}} \models \varphi$;
$\mathcal{U}, \overline{p_{ij}} \models$	$\psi; \langle \alpha \rangle \varphi$	*iff*	$\mathcal{U}, \overline{p_{iN}} \models \psi$, where $i \leq N < j$ and $\mathcal{U}, \overline{p_{Nj}} \models \langle \alpha \rangle \varphi$.

where $\pi^a \in \Pi$ and $\alpha, \beta \in \mathcal{A}_{\mathsf{SL}}$. ■

With these semantic rules, annotators can use formulae to determine if a property holds over a sequence of phonemes or not. Furthermore, they permit to alter their phonological

interpretation, depending on how the formula is build. For instance in the case where articulator a moves and articulator b remains static, and formula φ^a, giving information about articulator a exclusively, will see at least one Movement, whereas φ^b will see a single Hold. This is because the semantic rules rely on the individual articulation sequences to implicitly discard unused information; in this case, as formula φ^b only needed information from b, the channel of articulator a was ignored and the formula was satisfied directly on the individual articulation sequence $\hat{\mathcal{S}}^b$. This provides a mechanism to verify both individual and concurrent properties over the same structure.

4.4 Chapter conclusion

This chapter presented the main theoretical notions of this work. In the first part, a series of primitive sets where introduced, which serve as basis for the representation of SL phonetic parameters as logical statements (see Table 4.1). Additionally, a group of time-ordered data-structures was defined, in an effort to have a formal representation of temporal change. Both primitive sets and data-structures where connected with each other by way of a formal framework, based on the PDL, which was developed with the intention of being inherently multimodal, to cope with the notion of parallel articulatory changes.

Generally speaking, the main interest of the presented definitions is to establish a correspondence between very abstract notions and real world data. It is for this reason that primitive sets are introduced as templates to be filled by an expert, depicting fairly broad descriptions. The same holds true for the data-structures themselves, as they can correspond to different use cases depending on expert observations: for instance, the utterance objects from Definition 4.17 can serve to represent one-handed signs, two-handed signs, changes in NMFs and others.

The relationship between real-world observations and the formalism becomes clearer if we consider that a video from a corpus, for example, can be represented as an utterance object during an automatic recognition task; assuming that the proper technical tools are available, it is possible to follow changes on individual articulators and combine them on-the-fly, just as utterance objects do with timed articulation sequences. Furthermore, the flexibility of this particular data structure permits the addition or removal of channels as needed. In the real world, this can be useful for situations where a tracking system can only follow changes over a limited number of articulators, thus not needing to model any of the non-tracked information channels while still being interested on combining those which are.

Primitive Set	Description
$\mathcal{NAMES}$	The set containing every chain of characters.
$\mathcal{BODY}$	Contains any body part visible during signing, $\mathcal{BODY} \subset \mathcal{NAMES}$.
$\mathcal{ART}$	Refers to all body parts capable of articulation, $\mathcal{ART} \subseteq \mathcal{BODY}$.
$\Psi_{\mathtt{SL}}$	Establishes a three-dimensional coordinate system originating on the signer's center of mass, where $\Psi_{\mathtt{SL}} \subset \mathbb{R} \times \mathbb{R} \times \mathbb{R}$.
$\mathcal{LOC}$	Holds pairs of articulators with their respective places of articulation, denoted as (a, λ), where $a \in \mathcal{ART}$ and $\lambda \subset \mathcal{NAMES}$ (λ is the name of a location).
$\mathcal{CFG}$	Holds pairs of articulators with their possible configurations, denoted as (a, c), where $a \in \mathcal{ART}$ and $c \subset \mathcal{NAMES}$ (c is the name of a configuration).
Δ	Contains pairs of coordinates describing a vector in $\Psi_{\mathtt{SL}}$, $\Delta \subseteq \Psi_{\mathtt{SL}} \times \Psi_{\mathtt{SL}}$.
$\mathcal{L}_{\Delta}$	Refers to a function assigning names to vectors in Δ, permitting more abstract definitions such as "`Up`" or "`Down`".
$\mathcal{DIRECTIONS}$	It is the set of names that can be used by $\mathcal{L}_{\Delta}$ to label some vector, $\mathcal{DIRECTIONS} \subset \mathcal{NAMES}$.
$\Delta_{\mathtt{SL}}$	Contains sets of vectors identified by a individual direction names in $\mathcal{DIRECTIONS}$: for instance, it contains every individual vector that can be labeled as going "`Up`".
Π	Contains every atomic action related to every individual articulator in $\mathcal{ART}$.
Φ	Contains every atomic proposition related to every individual articulator in $\mathcal{ART}$.

Table 4.1: Summary of SL primitives and atoms for $\text{PDL}_{\mathtt{SL}}$

Something similar occurs with PDL_{SL} formulae: since they are intended to be interpreted over an utterance, they can be used to analyze SL videos once they have been turned into abstract representations. In particular, formulae can be used to ask question about the video, for instance, so as to determine whether a sign appears or not over a determined segment of video-frames. In this regard, the language is designed to be flexible enough to add or discard information depending on the available data, therefore adaptable to different types of corpora.

In order to give a better picture of how everything works in practice, the next chapter provides a concrete example. On Chapter 5, the formal structures defined on this chapter are used on a real-world recognition task, oriented towards the semi-automatic annotation of a corpus. Essentially, the idea is to show, step by step, how humans and an automatic system can interact to fill up the primitive sets (see Table 4.1), leading to the automatic specification of a corpus and its partial annotation by way of PDL_{SL} formulae.

Chapter 5

Semi-automatic annotation using logic

The previous chapter introduced the Propositional Dynamic Logic for Sign Language (PDL_{SL}), a formalism intended to represent Sign Language (SL) corpora in Natural Language Processing (NLP) tasks. Overall, the PDL_{SL} provides a set of basic rules to capture different corpus features under the same formal language, regardless of the technical characteristics of said corpus. For this reason, when confronted with real-world data, it is the task of the annotator to narrow down or extend the language as required, in order to suit his or her specific needs.

In this chapter, an example of the representation process is introduced with the implementation of a modular, PDL_{SL}-based, semi-automatic annotation framework. In particular, the system was designed to work over the Dicta-Sign corpus (see Chapter 3, Subsection 3.3.3, p. 65), a French Sign Language (LSF) corpus. Broadly, the system core consists on both an implementation of *utterances* (see Definition 4.17, p. 89), and a PDL_{SL} interpreter. A series of processing modules were built around these basic components, in order to automatically transform each corpus video into an instance of a utterance. Utterances are none other than directed acyclic graphs, linked to per-articulator Labeled Transition Systems (LTSs). In the final step, a series of PDL_{SL}-described properties are searched automatically in the graph, and the results are returned to the user as an ELAN annotation file.

In the following section, a brief description of the framework's architecture is presented, including a bird's-eye view of the overall process as intended in practice. Likewise, this portion contains a module-by-module description, demystifying the procedure on a step-by-step basis. Finally, the last couple of pages present the implementation of a semi-annotation system, created by following the described framework's guidelines.

5.1 Framework Architecture

As already stated, the main objective of the framework is to be able to receive an untreated SL video input, and return a set of possible annotations, calculated by way of formal verification. Figure 5.1 shows a diagram describing the data workflow.

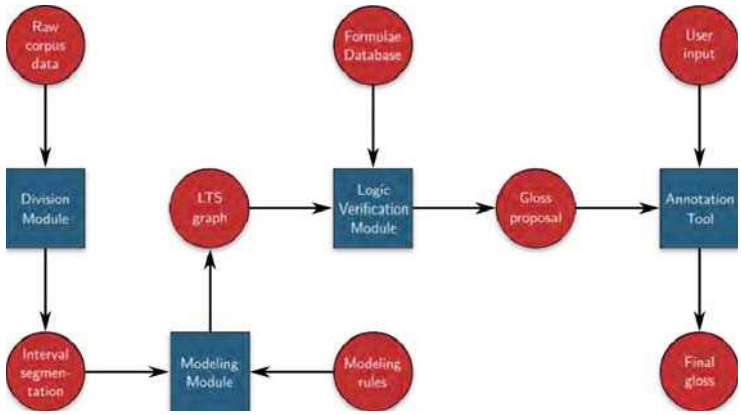


FIGURE 5.1: Block diagram of a generic PDL_{SL}-based SL lexical structure recognition system

In the Figure 5.1, circles represent information whereas squares depict individual processing modules; each module receives a particular input and their results are sent to other modules for further processing. Each module has its own specialized role, and together they take untreated corpus data and calculate an annotation file that can be used for linguistic research. In general, modules tasks can be broken as follows:

— **Division Module** Divides input data into time intervals.
— **Modeling Module** Encodes the information from time intervals into a graph.
— **Logic Verification Module** Uses the graph to verify a database of PDL_{SL} formulae, and generate an annotation.
— **Annotation Tool** Enables human experts to correct the results.

In the next sections, each of the modules depicted in the figure will be individually described, explaining in detail how each of them contributes to the generation of an automatic annotation.

5.1.1 Division Module

The *Division Module*, highlighted in Figure 5.2 is in charge of processing the *raw corpus data* and transform it into an *interval segmentation*.

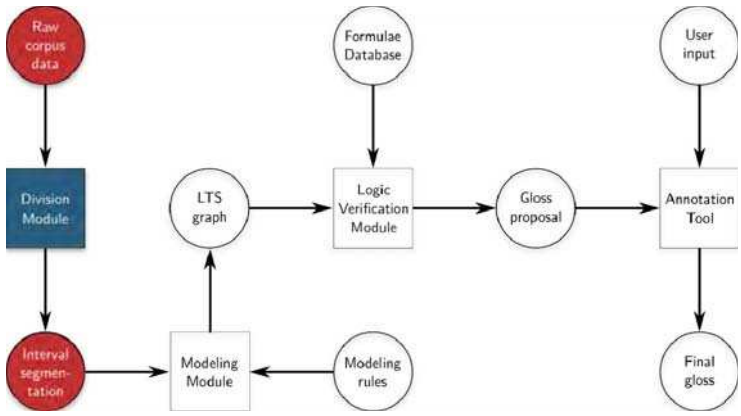


FIGURE 5.2: Block diagram of a generic PDL_{SL}-based SL lexical structure recognition system

The raw corpus data corresponds to untreated visual or numeric information, captured for the compilation of corpora. This can be anything from video-data, motion capture information, 3D depth maps of signers or any other technology used for the creation of a corpus. In the case of Dicta-Sign, raw data corresponds to 2D video files; that is, a series of time-ordered image frames, as depicted by Figure 5.3.

FIGURE 5.3: Example of a Dicta-Sign raw video sequence.

On the other hand, an interval segmentation consist of a separation of the raw data into time-segments, reflecting how signers' positions change in the video. Roughly, the module is in charge of finding contiguous frames where changes occur, obtaining an interleaving of change and static intervals, reminiscent of the Holds and Movements defined by the Movement-Hold Model (MHM); however, this segmentation may also contain noise intervals (*e.g.* changes without communication intent, like putting down the arms). Figure 5.4 shows an interval segmentation for the raw video.

FIGURE 5.4: Example of an interval segmentation for the right hand; the intervals below the image denote which frames are contained in the segment.

The segmentation shown in Figure 5.4, was built by looking for changes occurring over an individual articulator, in this case, the right hand. For each other articulator of interest the module will generate an individual segmentation. In this way, for the right and left hand there will usually be a pair of chains of intervals just like the ones depicted in Figure 5.5.

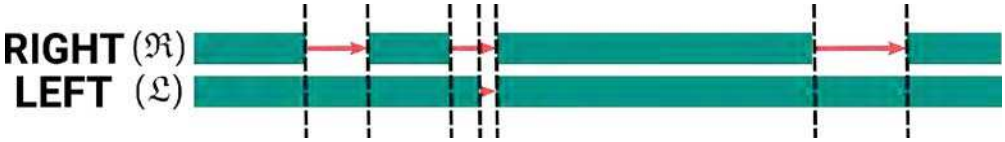


FIGURE 5.5: Example of two parallel per-articulator segmentations.

To treat videos from Dicta-Sign, the Division Module makes use of a tracking tool developed by [Gonzalez 2011] to obtain the 2D spatial positions of the hands and head of the signer. An example of the tracking results, overlaid over a video-frame, are shown in Figure 5.6.

FIGURE 5.6: Calculated path of hands and head on a corpus video.

On the figure, the points over the signer's hands and head represent the calculated positions for each. In addition, the colored line represents the path that the left hand followed when moving from one position to the other; the point where the line changes color, represents the instant where the hand achieved maximum speed.

Interval segmentation based upon this tracking tool, rely on defining articulation changes in terms of speed. In particular, the Division Module follows the observations made by [Gonzalez 2012], and selects segments by looking at speed variations. For the module, a frame is *static* if the speed of the observed articulator is below the noise threshold of the tracker (approx. 1.08 pixels/frame). Moreover, a frame is considered a *change* if its speed is above the threshold. On a similar note, static intervals are defined as sequences containing only static frames, while change intervals are sequences containing only change frames.

Since the tracking tool is limited to detect only hands and head, it is not possible to create a segmentation for any other articulators outside of these three. In addition, for Dicta-Sign in particular, the average speed of the head during movement held approximately the same value as the calculated noise threshold for the tested videos. A graphic showing how speed measures compare, for the three articulators, is presented in Figure 5.7. In the graph, speed changes below the noise threshold were set to zero, for clarity.

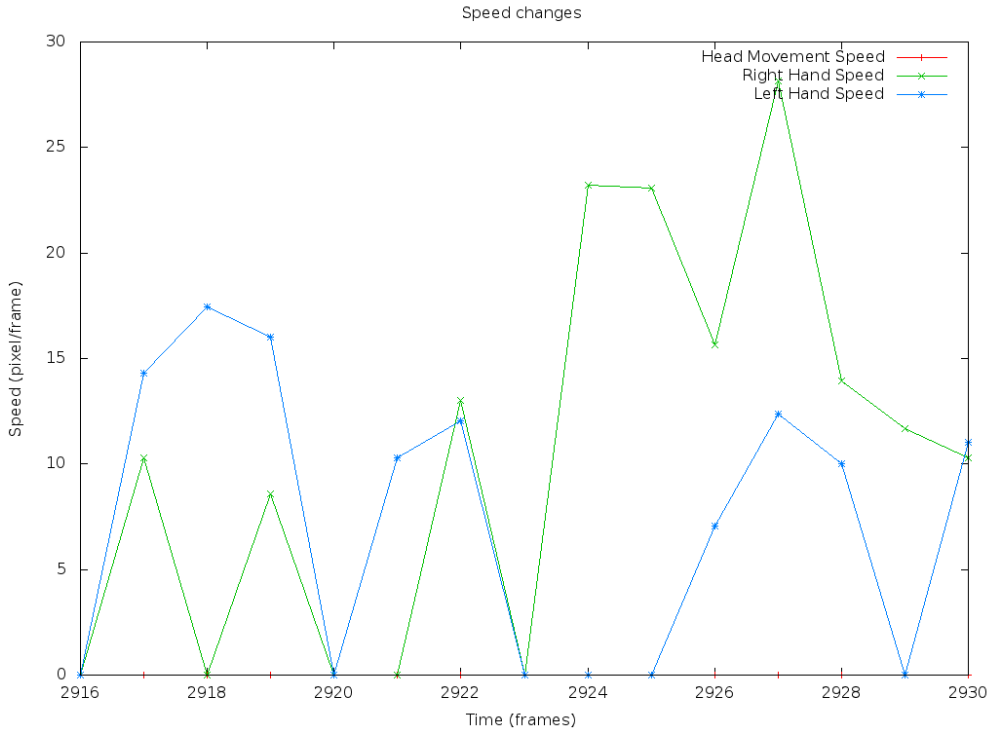


FIGURE 5.7: Example of an interval segmentation for the right hand; the intervals below the image denote which frames are contained in the segment.

Figure 5.7, shows a plot of per-articulator speed changes, occurring from frame 2916 to frame 2930 on a Dicta-Sign video. Observe that not only does the speed of the head remains fairly unchanged, but is always below the noise threshold (even in frames where both hands are moving). Under these conditions, the individual segmentation of the head is unreliable, hence forcing the module to only generate interval sequences for the hands. In addition, the detection of other change-inducing features, such as hand-configuration, isn't readily available. For this reason, segmentation has to be achieved by means of speed change alone, as presented by the Algorithm 1 below.

Recall from Subsection 4.3 (see p. 87) that intervals in this document are defined over the positive integers. Also, note that frames are enumerated over the same domain, thus starting in 1. Procedure SEGMENTATION receives the tracker's noise threshold T_ϵ and a total number of frames N. By definition, any speed lower than or equal to T_ϵ is assumed to be static. Also, the speed of any of the hands in the first frame (v_1) is 0, meaning that

Algorithm 1 Interval segmentation for a single hand in one video

```
1:  procedure SEGMENTATION(T_ε ≥ 0, N > 0)
2:      v_1 ← 0
3:      I ← {1}
4:      L ← empty set of intervals
5:      for each frame i ∈ {2, ..., N + 1} do
6:          v_i ← instantaneous hand speed at frame i
7:          if (v_i > T_ε & v_{i−1} ≤ T_ε) or (v_i ≤ T_ε & v_{i−1} > T_ε) then
8:              L.add(I)
9:              I ← {i}
10:         else
11:             I.add(i)
12:         end if
13:     end for
14:     L.add(I)
15:     return L
16: end procedure
```

the first frame is static by default. From there, for the hand of interest, at each frame $i > 1$, the algorithm reads the hand's speed v_i and uses it to decide whether i is part of an interval I or not; roughly, if i and $i-1$ are both of the same type and $i-1 \in I$, then $i \in I$. Finally, the procedure stops in $N+1$, to ensure that the sequence will always end on a static interval (we assume that $v_{N+1} = 0$).

Theorem 5.1 (Interleaved segmentation)**.** *The result of the execution of Algorithm 1, for any of the hands, is a list L either containing a series of interleaved static and change intervals or a single static interval.*

Proof. By induction over the length of L. Let I_0 be the first interval in L. Let i be the first frame where the hand changes, hence $v_i > T_\epsilon$ holds. If such i doesn't exist, every frame necessarily falls on the case of line 10, thus ensuring that they will all be added to interval I, defined in line 3. Since no change frame exist, I is static. Furthermore, because line 14 is always executed before returning, then I will be added to L, thus ensuring that $I = I_0$ and $L \neq \emptyset$.

If a change frame i does exist, we know from line 2 that $i \neq 1$. Also, note that for every frame $f \in [1, i-1]$ it holds that $v_f \leq T_\epsilon$, warranting that each will enter the case in line 10 and thus $I = [1, \ldots, i-1]$. Since $v_i > T_\epsilon$ and $v_{i-1} \leq T_\epsilon$, then i enters in the case of line 7, thus it holds that $L = [[1, \ldots, i-1]]$ and $I \leftarrow [i]$. In this scenario, it is always the case that $I = I_1$, since either it gets overridden again in line 9, after being added to L, or I is never overridden again and is added by line 14. Thus, for I_0 it holds.

For I_k, assume that $L = [I_0, \ldots, I_{k-1}]$ is correct. Let $I_{k-1} = [m - M, \ldots, m-2, m-1]$; interval I_{k-1} could have only been added to L by line 8, meaning that for frame m

it was the case that either ($v_m > T_\epsilon$ & $v_{m-1} \leq T_\epsilon$) or ($v_m \leq T_\epsilon$ & $v_{m-1} > T_\epsilon$). This situation renders frames $m-1$ and m of different types. In addition, we know that, by entering the case of line 7, frame m became the first frame in I, after adding I_{k-1} to L. Finally, from here, it will always be the case that $I = I_k$ since, once again, either I gets overridden in line 9 after being added to L, or I is never overridden again and is added by line 14. Thus, for I_k it holds. □

Algorithm 1 generates a valid sequence of interleaved change and static intervals; however, the changes inside the sequence aren't completely accurate in terms of what movement they are capturing. As observed by [Gonzalez 2012], it is often the case that signers chain hand movements without passing through a static interval, as they conserve momentum. An example can be seen in Figure 5.7 above: in frame 2926, the right hand ended some movement without attaining speed 0 and, immediately after, started a new one. This can be observed on the sudden drop of speed experienced by the hand, which is rapidly gained on the next instant. However, these changes remain largely invisible for Algorithm 1, which would consider $[2923, 2930]$ is a single change. To avoid this situation, change intervals are further subdivided with respect to speed minimums, by the procedure presented in Algorithm 2.

Algorithm 2 Change interval sub-segmentation for a single hand in one video

```
1:  procedure SubSegmentation(I_k = a change interval [m, n])
2:      if m = n or |I_k| < 3 then
3:          L_c.add(I_k)
4:      else
5:          I ← {m}
6:          L_c ← {}
7:          for each frame i ∈ {m + 1, ..., n − 1} do
8:              v_{i−1} ← instantaneous hand speed at frame i − 1
9:              v_i ← instantaneous hand speed at frame i
10:             v_{i+1} ← instantaneous hand speed at frame i + 1
11:             if (v_{i−1} > v_i & v_i < v_{i+1}) then
12:                 L_c.add(I)
13:                 L_c.add({i})
14:                 I ← {}
15:             else
16:                 I.add(i)
17:             end if
18:         end for
19:         I.add(n)
20:         L_c.add(I)
21:     end if
22:     return L_c
23: end procedure
```

Theorem 5.2 (Change interval subdivision)**.** *Executing Algorithm 2 over a change interval I_k, produces an interval chain L_c where, either $|L_c| = 1$ and $I_k \in L_c$ or $|L_c| \geq 3$ and L_c is an interleaving sequence of minimum speed intervals and change intervals.*

Proof. By induction, over the length if I_k.

If $|I_k| < 3$ then, by line 2 in the algorithm, it holds.

For $|I_k| = 3$, where $I_k = [i-1, i, i+1]$. By line 5, there is a variable I such that $I = \{i-1\}$. The **for** of line 7 centers in i and compares the speeds of each frame denoted by I_k; v_{i-1}, v_i and v_{i+1}.

If $v_{i-1} > v_i < v_{i+1}$, then, by lines 12–14, we know that $L_c = (\{i-1\}, \{i\})$ and $I = \emptyset$. With $|I_k| = 3$, the **for** is executed only once, thus, the algorithm executes lines 19 and 20, resulting in $L_c = (\{i-1\}, \{i\}, \{i+1\})$, which is returned immediately after. Note that, by the sake of the comparison in line 11, $\{i\}$ denotes a minimum. Then, for $|I_k| = 3$, L_c holds.

On the other hand, if $v_{i-1} > v_i < v_{i+1}$ is false, then I doesn't change, thus $I = \{i-1\}$. By line 16, $I = \{i-1, i\}$. Once again, the **for** is executed only once, thus, the algorithm executes lines 19 and 20, giving $I = \{i-1, i, i+1\}$, $L_c = (I)$ and returning L_c. Since $I \equiv I_k$, it holds.

For $|I_k| = N > 3$. Assume without loss of generality that $I_k = \{0, 1, \ldots, N\}$. Also, assume that, until $N-1$, L_c is a correct interleaving. Recall that the **for** ranges from $1, \ldots, N-1$. In the last execution, it must have compared speeds v_{N-2}, v_{N-1} and v_N.

If $v_{N-2} > v_{N-1} < v_N$, then $L_c = (\ldots\{\ldots N-2\}, \{N-1\})$ where $\{\ldots N-2\}$ is a change and $\{N-1\}$ is a minimum. Note that the comparison ensures that $I = \{\ldots N-2\}$ will always be a change, as in the previous **for** execution, line 11 tested $v_{N-2} > v_{N-1}$ and added $N-2$ to I. Now, as $\{N-1\}$ was the last interval added to L_c, then $I = \{\}$. Also, since $N-1$ corresponds to the last test of the **for**, the algorithm executes lines 19 and 20 giving $I = \{N\}$ and $L_c = (\ldots, \{\ldots N-2\}, \{N-1\}, \{N\})$ is returned. Since we know that, $v_{N-1} < v_N$, then N is not a minimum, therefore L_c holds.

If not $v_{N-2} > v_{N-1} < v_N$, then by line 16 we have that $I = \{\ldots, N-1\}$. Since this is the last execution of the **for**, the algorithm executes lines 19 and 20 giving $I = \{\ldots, N-1, N\}$ and $L_c = (\ldots, \{\ldots N-1, N\})$ is returned. Note that, in this case, neither $N-1$ nor N can be minimums, since frame $N+1$ is static and, hence, $v_N > v_{N+1} = 0$. Thus L_c holds. □

The previous theorem, shows that Algorithm 2 uses speed minimums to produce an interleaving, depicting similar characteristics to those resulting from the first algorithm. In turn, this similarity can be exploited to combined both results into a single interval chain: the idea is to substitute all changes in the first sequence with their sub-segmentation, if any. In general, this corresponds to forcing local minimums into static intervals, a process presented in Algorithm 3.

Algorithm 3 Substitution of change intervals

```
1:  procedure SegmentSubstitution(L = interval sequence)
2:      if |L| == 1 then
3:          L_s ← L
4:      else
5:          L_s ← {}
6:          for each interval I ∈ L do
7:              if I is a static interval then
8:                  L_s.add(I)
9:              else
10:                 for each subinterval I_s ∈ SubSegmentation(I) do
11:                     L_s.add(I_s)
12:                 end for
13:             end if
14:         end for
15:     end if
16:     return L_s
17: end procedure
```

Theorem 5.3 (Change interval substitution)**.** *In the resulting interval list L_s, produced by Algorithm 3, every "true" change interval will be surrounded by either two static intervals, two minimum speed intervals or one of each.*

Proof. Assuming that L is a correct interleaving, if $|L| = 1$ then there are no change intervals, so it holds by vacuity. For $|L| > 1$, we know by Theorem 5.1 that every change interval in L is surrounded by two static intervals; recall that L always ends on a static interval. By Theorem 5.2 we know that SubSegmentation(I) in line 10 is an interleaving of change and minimum speed intervals, where the first and last $I_s \in$ SubSegmentation(I) are always "true" changes. Thus, for $|\text{SubSegmentation}(I)| \leq 3$ it holds: the first and last change intervals will be surrounded by one minimum speed and one static interval, while any other will be surrounded by minimum speed intervals. □

Even though, by definition, minimum speed intervals are not truly static, the system makes no difference between the two; to avoid any potential pitfalls in this regard (at least from the implementation point of view), each time a minimum frame is found the module sets its speed to zero, so as to maintain consistency. An example of the entire process, for the right hand, is given on Figure 5.8.

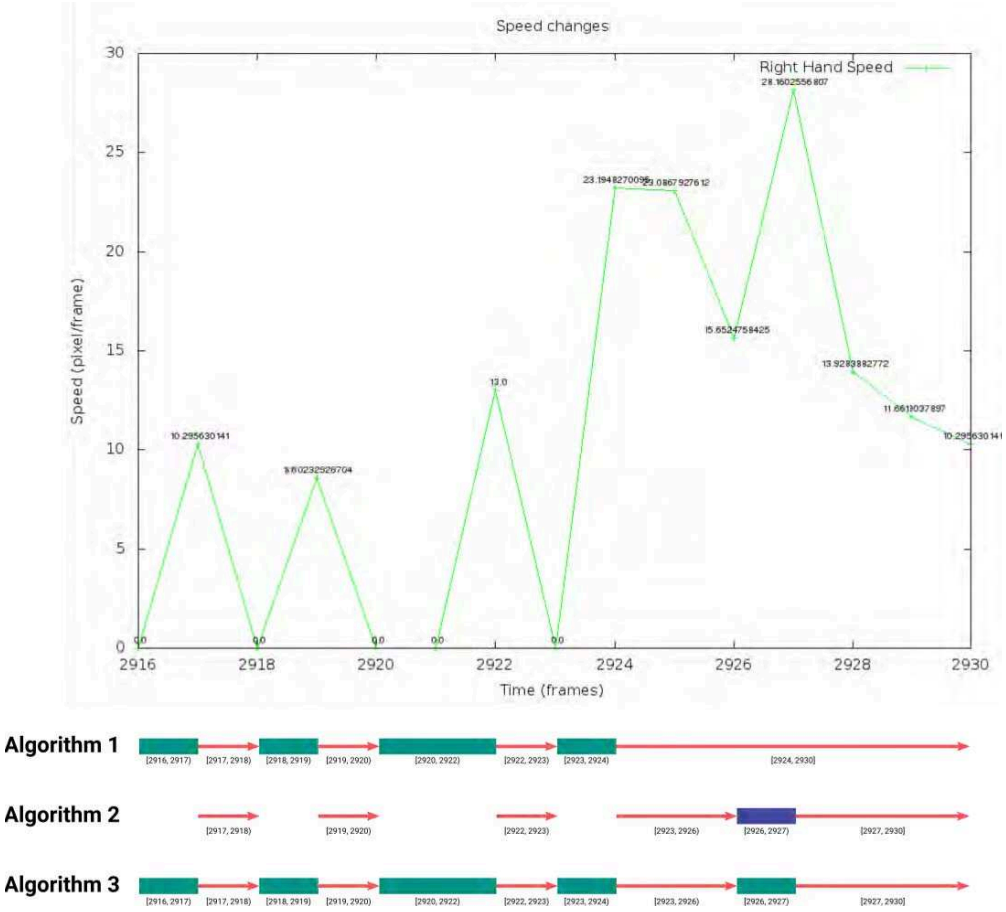


FIGURE 5.8: Example of an interval segmentation for the right hand. The intervals sequences below the graph denote the results produced by each algorithm. The graph depicts the speed changes of the right hand, over a video segment.

In Figure 5.8, a speed graph for the right hand is presented, with its corresponding interval segmentation; there, intervals are depicted as either boxes (static) or arrows (change), depending on how they are interpreted by the module. Broadly, the first segmentation presents the resulting sequence after executing Algorithm 1. The second, presents how changes are affected when passed through Algorithm 2. Finally, the last sequence presents the result of combining the two previous ones, by way of Algorithm 3; this is the one that will be passed over to the next module, for further processing. Also, note that the depicted sequences are just segments of a larger sequence, produced for the entire video. It is for this reason that the first sequence ends on a change interval and not on a static one (more intervals follow).

The choice of introducing artificial static intervals, was made to capture instants within long movement chains where it is clear that more than one change occurred. Further down the line, these reduced states serve as points of reference for the verification module, to detect smaller changes within these movement chains. Also, note that these points contain exactly one frame, as the comparisons in line 11 of Algorithm 2 are strict. This

is to avoid, in the measure of possibility, breaking up changes that may be intentionally slower.

The module executes Algorithm 3 for each hand and passes both results to the next module, where they will be formally represented as explained in the next section.

5.1.2 Modeling Module

The *Modeling Module*, shown in Figure 5.9, is the one in charge of transforming the raw video information into a formal data structure. More precisely, the module receives an interval segmentation for every articulator, like the one shown in Figure 5.5, and uses the available information to determine the truth value of a series of statements about articulators.

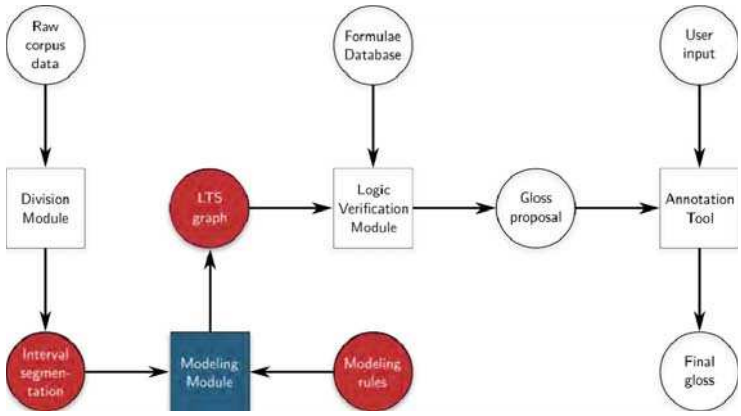


FIGURE 5.9: Block diagram of a generic PDL_{SL}-based SL lexical structure recognition system

More specifically, this module implements the utterance objects presented in Definition 4.17, and executes an algorithm to tag utterances with information, depending on what it can automatically infer from video images. For this reason, in order to generate the graph, the module has to know how to make sense of visual information, and convert it into a more abstract representation. This is achieved by giving concrete domains to the PDL_{SL} primitive sets of Section 4.1 (p. 79); once this information is in place, the module becomes knowledgeable enough to determine which articulators are being described, which statements are true on a given video-frame and, overall, the kind of data that has to be modeled.

In the next section, primitive information is defined for the case of Dicta-Sign corpus, when paired with the tracker presented by [Gonzalez 2011].

5.1.2.1 Primitives for a 2D corpus

Recall from last chapter (see Chapter 4, Section 4.1 p. 79) that PDL_{SL} defines a series of set templates, intended to be used in the creation of atomic propositions and actions; originally, these sets where defined very broadly, as they have to be general enough to cover a vast number of use cases. In the case of the study of a specific corpus, as is the case with Dicta-Sign, it is not necessary to have such broad definitions. In fact, confronted with data, the opposite becomes true: it is necessary to have a very detailed account of what is to be included in each of these sets. The existing templates are explained briefly in Figure 5.10.

$\mathcal{BODY}$	=	the set of relevant body parts (Definition 4.1, p.79)
$\mathcal{ART}$	=	the set of SL articulators (Definition 4.2, p.79)
Ψ_{SL}	=	a set of euclidean coordinates used in SL (Definition 4.3, p.80)
$\mathcal{LOC}$	=	the set of places of articulation (Definition 4.4, p.81)
$\mathcal{CFG}$	=	the set of articulator configurations (Definition 4.5, p.81)
Δ_{SL}	=	a set of directions used to model position and movement (Definition 4.8, p.82)

FIGURE 5.10: Primitive set templates in PDL_{SL}

Each of these sets have to be adapted to work over Dicta-Sign. First of all, from Definition 4.1 recall that the set $\mathcal{NAMES}$ contains all possible chains of characters; for implementation purposes, $\mathcal{NAMES}$ corresponds to any string representation on any programming language. For Dicta-Sign, the set of *relevant body parts* $\mathcal{BODY} \subseteq \mathcal{NAMES}$ will be redefined as follows:

Definition 5.1 (Relevant body parts for 2D corpora)**.**

$$\mathcal{BODY} = \{\mathfrak{R}, \mathfrak{L}, \texttt{HEAD}, \texttt{UPPERBODY}, \texttt{LOWERBODY}\}$$

where $\mathfrak{R}$ and $\mathfrak{L}$ denote the right and left hand, respectively. ■

The set $\mathcal{BODY}$ contains names for visible body parts in Dicta-Sign corpus; the parts were selected from what is conceivably recognizable by exploiting the tracker by [Gonzalez 2011]. Also, recall that the set of articulators $\mathcal{ART}$ is a subset of $\mathcal{BODY}$, thus giving the following:

Definition 5.2 (Articulators for 2D corpora)**.**

$$\mathcal{ART} = \{\mathfrak{R}, \mathfrak{L}\}$$

■

Recall that, because of the characteristics of the tracking algorithm, the Division Module only obtains interval segmentations from the hands; in essence, this means that the only articulation information passed to the Modeling Module comes from the hands, thus limiting the contents of $\mathcal{ART}$.

With respect to space, it has already been mentioned that Dicta-Sign is a 2D corpus; for this reason, the coordinate system established by $\Psi_{\texttt{SL}}$ is limited to two dimensions, simply giving:

Definition 5.3 (Space coordinates for 2D corpora)**.**

$$\Psi_{\texttt{SL}} = \mathbb{R} \times \mathbb{R} \times \{0\}$$

■

In general, $\Psi_{\texttt{SL}}$ doesn't change with respect to Definition 4.3; it is a given that the only possible value in the third dimension in Dicta-Sign, reachable by any articulator, is zero. Hence, the original definition is valid and compatible with Definition 5.3.

Once articulators and their spatial frame of reference is determined, it is possible to define their places of articulation. From Definition 4.4, recall that each articulator $a \in \mathcal{ART}$ will have its own set of places $\mathcal{LOC}^a$. In the case of the hands, both sets will almost be identical, with the exception that they won't contain themselves as locations.

Definition 5.4 (Places of articulation for 2D corpora)**.** Recall that $\mathcal{BODY}^a$ denotes the containing body-bound locations where a can articulate. For articulators $\mathfrak{R}, \mathfrak{L} \in \mathcal{ART}$, it is the case that:

$$\mathcal{BODY}^{\mathfrak{L}} = \mathcal{BODY}/\{\mathfrak{L}\}$$

$$\mathcal{BODY}^{\mathfrak{R}} = \mathcal{BODY}/\{\mathfrak{R}\}$$

Recall the labeling $\mathcal{C} : 2^{\Psi_{\texttt{SL}}} \rightarrow \mathcal{NAMES}$, mapping space regions to names, from Definition 4.4. The inverse function, $\mathcal{C}^{-1} : \mathcal{NAMES} \rightarrow 2^{\Psi_{\texttt{SL}}}$, takes members of $\mathcal{NAMES}$ to space regions. Let the set $\mathcal{REGIONS} = \{\mathcal{C}^{-1}(\texttt{NEUTRAL}), \mathcal{C}^{-1}(\texttt{RIGHT\_HEADSIDE}), \mathcal{C}^{-1}(\texttt{LEFT\_HEADSIDE})\}$ denote a group of spatial regions. Articulation places are given as follows:

$$\mathcal{LOC}^{\mathfrak{L}} = \{\lambda : \lambda \in (\mathcal{BODY}^{\mathfrak{L}} \cup \bigcup_{r \in \mathcal{REGIONS}} \mathcal{C}(r))\}$$

$$\mathcal{LOC}^{\mathfrak{R}} = \{\lambda : \lambda \in (\mathcal{BODY}^{\mathfrak{R}} \cup \bigcup_{r \in \mathcal{REGIONS}} \mathcal{C}(r))\}$$

$\mathcal{LOC}$ remains the same as in Definition 4.4. ■

The visual depiction of the defined places of articulation is shown in Figure 5.11; each region can be as small or as large as desired, depending on the definition of $\mathcal{C}^{-1}$.

FIGURE 5.11: Possible places of articulation in $\mathcal{LOC}$ common to both hands.

Note that the defined regions may overlap. Because of this, in some video corpora there might be ambiguity in region definitions, due to the lack of a third dimension: in Figure 5.11, the neutral space should denote the 3D spatial coordinates around the signer, which is not possible to model in a 2D corpus. In this context, `UPPERBODY` and `LOWERBODY` are inaccurately defined as sub-planes of `NEUTRAL`, because of this technical limitation. Nevertheless, the advantage of holding the meaning of spatial locations into an external function, in this case $\mathcal{C}^{-1}$, is that it can be easily adapted for the 3D case by changing the function itself, while leaving the region names intact.

Definition 5.4 is largely based on function $\mathcal{C}$; from the implementation point of view, this corresponds to a data structure, mapping 2D image subsections to strings. For Dicta-Sign, the human annotator can define regions as rectangles on the image's coordinate system, thus permitting to easily add or remove articulation places as needed; in this sense, it is the internal representation itself who provides $\mathcal{C}$ rather than the user.

The last set defined on a per-articulator basis, is the one containing the possible configurations an articulator can assume. In this case, as the only articulators in $\mathcal{ART}$ are the hands, the configuration set $\mathcal{CFG}$ will contain only hand-configurations. However, as previously explained, the used tracking tool is unable to detect complex features such as configuration change; because of this, the module isn't capable of using $\mathcal{CFG}$. Regardless, it is useful to provide these definitions anyway, at least to exemplify how they fit in the overall process. The proposed $\mathcal{CFG}$ is presented on Definition 5.5 below.

Definition 5.5 (Articulator configurations for 2D corpora)**.** For $\mathfrak{R}, \mathfrak{L} \in \mathcal{ART}$, configurations are given as follows:

$$\mathcal{CFG}^{\mathfrak{R}} = \mathcal{CFG}^{\mathfrak{L}} = \{\texttt{INDEX}, \texttt{PINCE}, \texttt{FIST}, \texttt{PALM}, \ldots\}$$

$\mathcal{CFG}$ remains the same as in Definition 4.5. ■

In general, the names included in $\mathcal{CFG}$ correspond each to a possible hand-configurations in LSF, some of which are depicted in Figure 5.12. LSF hand-configurations where chosen because the tests were based on the LSF portion of Dicta-Sign. Nevertheless, just as with the other sets, the contents of the set largely depend on the annotator, and are by no means limited by what is shown on the figure.

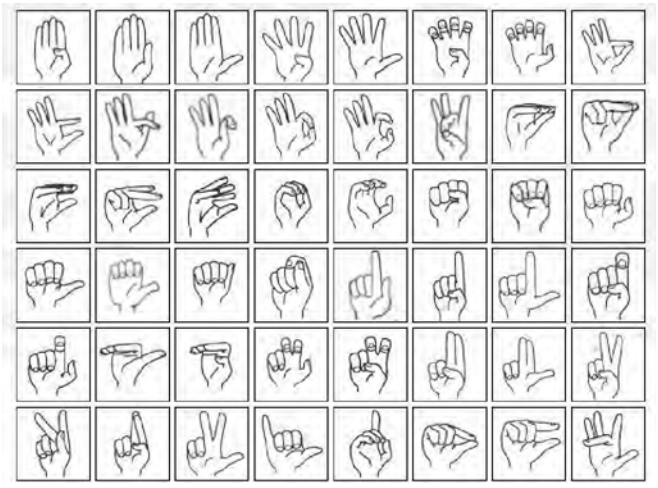


FIGURE 5.12: Some LSF hand configurations.

Finally, in order to give precisions about movement direction or other spatial relations, the module fills the set of directions with 2D vectors as shown in Definition 5.6.

Definition 5.6 (Set of relevant directions for 2D corpora)**.** Recall from Definition 4.7 the set of direction names $\mathcal{DIRECTIONS} \subset \mathcal{NAMES}$. Roughly, each member in $\mathcal{DIRECTIONS}$ will be a string identifying a family of vectors, characterized by the mapping $\mathcal{L}_\Delta : 2^\Delta \to \mathcal{DIRECTIONS}$, where Δ is a set of vectors in $\Psi_{\texttt{SL}}$. For simplicity, for the 2D case the selected names are simply arrows representing the direction of the movement:

$$\mathcal{DIRECTIONS} = \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow\}$$

The set of relevant directions is defined as follows:

$$\Delta_{\texttt{SL}} = \bigcup_{t \in \mathcal{DIRECTIONS}} \mathcal{L}_\Delta^{-1}(t)$$

where $\mathcal{L}_{\Delta}^{-1}(t)$ is defined for every $t \in \mathcal{DIRECTIONS}$. ■

Similar to what happens with the places of articulation, $\Delta_{\mathtt{SL}}$ is completely based on how the function $\mathcal{L}_{\Delta}$ was defined. For its implementation, the definition comes rather straightforward: for example, $\leftarrow$ are all vectors with starting and ending coordinates $(start_0, start_1)$ and (end_0, end_1), respectively, such that $start_1 \equiv end_1$ and $start_0 < end_0$. This is, they only change horizontally, towards the left side of the coordinate system. For the rest of the names in $\mathcal{DIRECTIONS}$, similar definitions were made in $\Delta_{\mathtt{SL}}$. Note that, by convention, directions are interpreted from the signer's point of view; also, $\Psi_{\mathtt{SL}}$ is centered on the signer's body. An diagram exemplifying the selected point of view is shown on Figure 5.13.

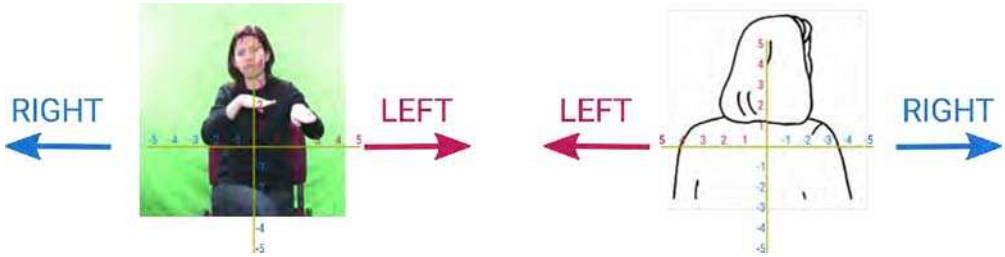


FIGURE 5.13: The signer's point of view, used to define $\Psi_{\mathtt{SL}}$.

With this last definition, the contents of the primitive sets (see Chapter 4, Section 4.1, p. 79) have been completely specified for the case of a 2D corpus. From here, it becomes possible to define the sets of atomic propositions and actions, by using the members of the sets above to give statements about video segments. This process is presented in the next section.

5.1.2.2 Atomic propositions and actions for a 2D corpus

As presented in Chapter 4 (see Section 4.2 p. 83), atomic actions and propositions are the basis of what is representable with $\mathrm{PDL}_{\mathtt{SL}}$. In general, they correspond to statements asserting information about the state of a signer during signing. However, even though the Modeling Module knows how atoms combine, it can't determine by itself which assertions are meaningful for the studied corpus. For this reason, it corresponds to the annotator to determine what should be claimed, depending on the available tools.

Remember from the previous section that, for the time being, only the hands are considered articulators. The atomic propositions for hand $a \in \mathcal{ART}$ are defined below.

Definition 5.7 (Atomic propositions for 2D corpora)**.** The set Φ^a of *atomic propositions for articulator* $a \in \mathcal{ART}$ is defined as:

$$\Phi^a = \{\mathbf{pos}(a, \delta, b), \mathbf{loc}(a, \lambda), \mathbf{touch}(a, b), \mathbf{cfg}(a, c)\}$$

where $a, b \in \mathcal{ART}$, $\delta \in \mathcal{DIRECTIONS}$, $\lambda \in \mathcal{LOC}^a$ and $c \in \mathcal{CFG}^a$.

Intuitively:

— $\mathbf{pos}(a, \delta, b)$.- Recall the function $\mathcal{L}_\Delta$ from Definition 4.7. This atom asserts that hand a is located on position δ with respect to b; this is, there exist a vector $\overrightarrow{BA} \in \mathcal{L}_\Delta^{-1}(\delta)$ such that a is positioned in coordinate A and b is positioned in coordinate B.

— $\mathbf{loc}(a, \lambda)$.- Indicates that hand a articulated in location λ.

— $\mathbf{touch}(a, b)$.- Is the same as $\mathbf{loc}(a, b)$, where b is a hand such that $b \in \mathcal{LOC}^a$ and $b \in \mathcal{ART}$.

— $\mathbf{cfg}(a, c)$.- Hand a holds configuration c.

The set Φ is the same as in Definition 4.9. ■

An example of how atomic propositions assert information about a video segment, is presented in Figure 5.14.

FIGURE 5.14: The information about the signer's posture above is captured by the following atomic propositions, which are true for this image: $\mathbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L})$, $\mathbf{pos}(\mathfrak{L}, \rightarrow, \mathfrak{R})$, $\mathbf{loc}(\mathfrak{R}, \texttt{HEAD})$, $\mathbf{loc}(\mathfrak{L}, \texttt{HEAD})$, $\mathbf{cfg}(\mathfrak{R}, \texttt{CLAMP})$, $\mathbf{cfg}(\mathfrak{L}, \texttt{CLAMP})$, $\mathbf{touch}(\mathfrak{R}, \mathfrak{L})\ldots$

The atomic propositions in Definition 5.7, assert information occurring when the signer holds his or her posture without moving, as presented in Figure 5.14. However, propositions alone are not enough to give information about change; this is done by way of the set of atomic actions, provided by Definition 5.8.

Definition 5.8 (Atomic actions for 2D corpora)**.** The set Π^a of *atomic actions for articulator* $a \in \mathcal{ART}$ is defined as:

$$\Pi^a = \{\delta_a, \circlearrowleft_a, \circlearrowright_a, \leftrightsquigarrow_a\}$$

where $\delta \in \mathcal{DIRECTIONS}$ and $a \in \mathcal{ART}$.

Intuitively:

— δ_a.- Indicates that hand a moves in direction δ (*i.e.* $\rightarrow_a$ asserts that hand moves to the signer's left).

— $\circlearrowleft_a$.- Asserts that a executes a single clockwise-motion.

— $\circlearrowright_a$.- Similarly, asserts that a executes a single counterclockwise-motion.

— $\leftrightsquigarrow_a$.- The hand a does a *trill* motion: it moves rapidly and continuously without changing its articulation place.

The set Π is the same as in Definition 4.10. ■

Actions assert information about change. An example of this is presented in Figure 5.15, where atoms $\leftarrow_{\mathfrak{R}}$ and $\rightarrow_{\mathfrak{L}}$ capture the changes occurring in the right and left hand, respectively, during some interval of time.

FIGURE 5.15: Atomic actions $\leftarrow_{\mathfrak{R}}$ and $\rightarrow_{\mathfrak{L}}$ are true in the above image.

In contrast to the members of set $\Delta_{\mathtt{SL}}$, the movements $\circlearrowleft_a$, $\circlearrowright_a$ and $\leftrightsquigarrow_a$ in Definition 5.8 are not thoroughly defined. This is because it is assumed that they can be approximated by combining small chains of movements in $\Delta_{\mathtt{SL}}$: for example, $\circlearrowright_a$ could be seen as a sequence $(\nearrow, \rightarrow, \searrow, \downarrow, \ldots)$. In this regard, it is important to underline that atomic actions happen over a single change interval: they are swift, uncut movements.

Finally, it is important to underline that the definitions of Φ and Π remain unchanged. With this, the Modeling module can directly create and label utterance objects, by way of a single algorithm. The details of this process are presented in the next section.

5.1.2.3 Transforming interval segments into utterance objects

Recall that the Division Module results into a segmentation of change and static intervals, as depicted in Figure 5.16.

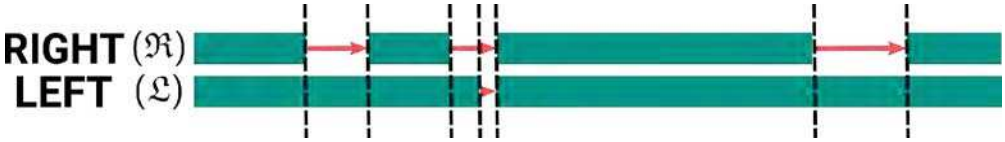


FIGURE 5.16: Articulator segmentations for the right and left hands.

The Modeling Module, is in charge of transforming each of these segmentations into timed articulation sequences $\hat{\mathcal{S}}^a$, the ones presented in Definition 4.15. Remember that a timed articulation sequence is none other than a chain of transitions, induced by some articulation model $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ (see Definition 4.13).

Intuitively, the articulation model of articulator a, encodes all possible significant postures of a as a LTS. For instance, the articulation model of the right hand $\mathcal{M}^{\mathfrak{R}}$ (in LSF), has some state s_0 where $\mathbf{cfg}(\mathfrak{R}, \texttt{INDEX})$ is true (the right hand has the index finger extended). If there are n possible configurations in $\mathcal{CFG}^a$, then there are at least $n-1$ transitions starting in s_0, taking the hand from `INDEX` to any other valid configuration.

However, articulation models are not a given for the module; each interval in the input segmentation is interpreted a valid state of some unknown model. Thus, the system is incapable of asserting information about the validity of each $\mathcal{M}^a$, but may infer its states, induced by static intervals, and how they connect with each other. To this end, the system needs a way to assert the propositional information on each state; a method to find $[\![\cdot]\!]_{\Phi^a}$ and $[\![\cdot]\!]_{\Pi^a}$. This is done by way of *modeling rules*, as defined below.

Definition 5.9 (Modeling rule)**.** Let Φ and Π be the sets of atomic propositions and actions. A *modeling rule* is a function $\mathcal{R} : (\Phi \cup \Pi) \times (\mathbb{N} \times \mathbb{N}) \to \{\top, \bot\}$, taking proposition and time intervals to truth values. ■

From the implementation point of view, modeling rules are functions that indicate whether an atom holds over some video segment. Intuitively, they are the rules used to determine the meaning of atomic actions and propositions with respect to real-world data.

As with the labelings for regions, configurations and directions, these functions are not directly implemented by the module, but are provided externally. For this particular case, the existing rules are able to determine the truth values of simple statements over 2D images, using only hand tracking; this is, whether a hand is inside some region (for

pos), if it is located in certain position with respect to the other (for **touch**), if there is a clear direction of movement (for δ_a), etc. In turn, these rules enable the construction of model $\mathcal{M}^a$, sequence $\hat{\mathcal{S}}^a$ and, ultimately, utterance object $\mathcal{U}$ (see Definition 4.17). The process is detailed by Algorithm 4.

Algorithm 4 Transformation of interval segments into utterance objects

procedure UTTERANCECONSTRUCTION($L_0, \ldots, L_n$, $\mathcal{R}$)
  $\mathcal{SEQ} \leftarrow \{\}$
  for each segmentation list $L_a \in (L_0, \ldots, L_n)$ **do**
    let $\mathcal{M}^a = (S, R, [\![\cdot]\!]_{\Phi^a}, [\![\cdot]\!]_{\Pi^a})$ be an empty articulation model of a.
    let $\hat{\mathcal{S}}^a = (\mathcal{S}^a, \mathcal{T}^a)$ be an empty timed articulation sequence of a.
    for each change interval $I_i \in L$ **do**
      let I_{i-1} and I_{i+1} be the static intervals before and after I_i
      create two states s_{i-1} and s_{i+1} in $\mathcal{M}^a$
      assert $(s_{i-1}, s_{i+1}) \in R$ in $\mathcal{M}^a$
      for each atom proposition $p^a \in \Phi^a$ **do**
        if $\mathcal{R}(p^a, I_{i-1})$ **then**
          assert $p^a \in [\![s_{i-1}]\!]_{\Phi^a}$
        end if
        if $\mathcal{R}(p^a, I_{i+1})$ **then**
          assert $p^a \in [\![s_{i+1}]\!]_{\Phi^a}$
        end if
      end for
      for each atom action $\pi^a \in \Pi^a$ **do**
        if $\mathcal{R}(\pi^a, I_i)$ **then**
          assert $\pi^a \in [\![(s_{i-1}, s_{i+1})]\!]_{\Pi^a}$
        end if
      end for
      append (s_{i-1}, s_{i+1}) to $\hat{\mathcal{S}}^a$
      assert $I_i \in \mathcal{T}^a((s_{i-1}, s_{i+1}))$ in $\hat{\mathcal{S}}^a$
    end for
    $\mathcal{SEQ}$.add($\hat{\mathcal{S}}^a$)
  end for
  return utterance $\mathcal{U}$ created from $\mathcal{SEQ}$
end procedure

Theorem 5.4 (Modeling Complexity)**.** *Creation of the $\mathcal{U}$ object takes $\mathcal{O}(n)$ time.*

Proof. For the n detected change interval, from every articulator, the algorithm:

- adds two vertex to the graph $\mathcal{M}^a$ in $\mathcal{O}(1)$ time;
- adds an edge to the graph $\mathcal{M}^a$ in $\mathcal{O}(1)$ time;
- performs a constant number of comparisons, for the members of Π and Φ in $\mathcal{O}(1)$ time;
- performs one insertion to a list $\hat{\mathcal{S}}^a$ in $\mathcal{O}(1)$ time;
- performs one insertion to a lookup table $\mathcal{T}^a$ in $\mathcal{O}(1)$ time;

thus the complexity is roughly $\mathcal{O}(n)$.

Finally, from Lemma 4.5, utterance labeling takes up to $\mathcal{O}(2n+1)$ time, thus the overall complexity remains bounded by $\mathcal{O}(n)$. □

Algorithm 4, receives a set of interval sequences for n articulators, and a modeling rule $\mathcal{R}$. For each interval in each sequence, it infers the information on the model by using $\mathcal{R}$ to test, one by one, each of the defined atoms. Simultaneously it builds the set $\mathcal{SEQ}$, which can be used to generate utterance $\mathcal{U}$ in line 28. Also, note that $\mathcal{U}$ is well-defined; from Definition 4.17, it is clear that the set $\mathcal{SEQ}$ is the only external requirement to build $\mathcal{U}$, thus no further procedures are needed. Figure 5.17 shows a visual representation of the results.

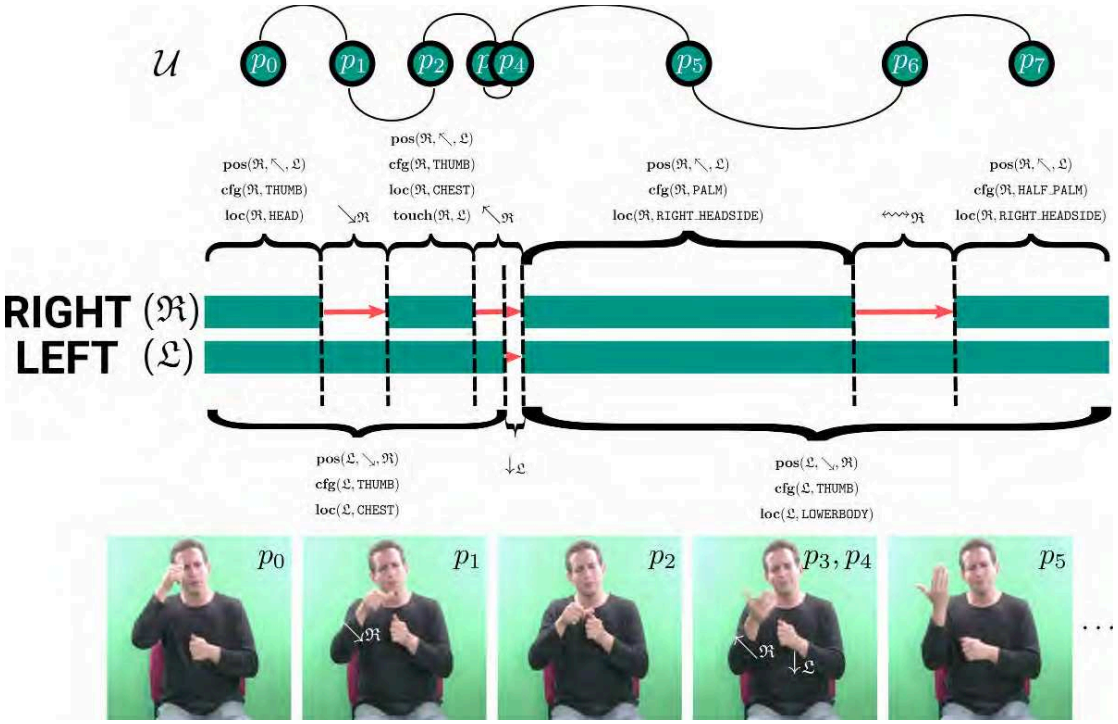


FIGURE 5.17: Transformation of interval segments into utterance phonemes.

Figure 5.17 presents the induced utterance object for two interval segmentations. In the two sequences, each of the intervals is marked with the propositional information inferred by the given modeling rules. The end result, the utterance object $\mathcal{U}$ above, is time-aligned with interval changes, as it was created from two timed articulation sequences based on the given segmentation; in essence, the depicted graph corresponds to the abstract representation of a video-segment, as shown on the bottom of the image. A simplified representation of the results of both the Division and Modeling modules together is shown in Figure 5.18.

Once the utterance construction is complete, the resulting graph is passed to the next module, which will be in charge of analyzing the result by way of formal verification. The process is explained in the next section.

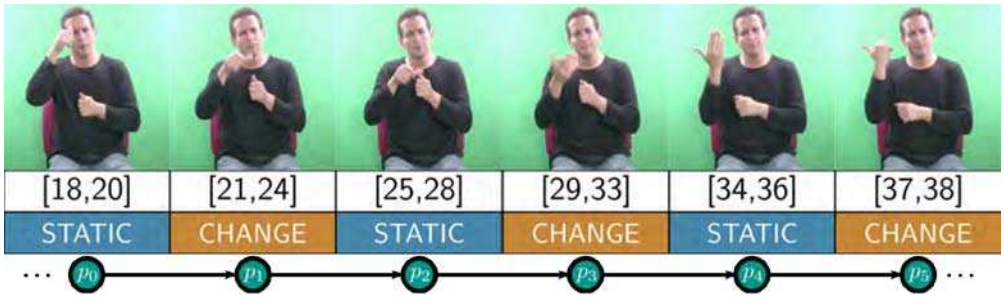


FIGURE 5.18: Example of the different layers processed by an automatic annotation system

5.1.3 Logic Verification Module

The *Logic Verification Module* is the one in charge of taking the graph generated by the Modeling module, in order to verify if a database of PDL_{SL} formulae hold over it. Broadly, each formula corresponds to the description of either a sign or a number of phonetic level properties. A diagram showing the place of the module in the framework is given in Figure 5.19.

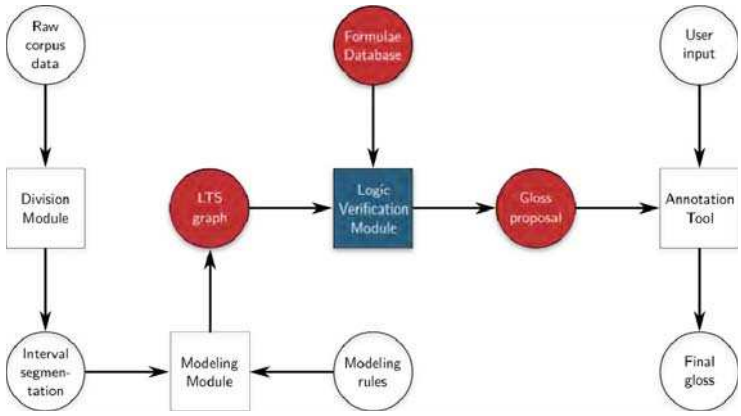


FIGURE 5.19: Block diagram of a generic PDL_{SL}-based SL lexical structure recognition system

The input of the module is the utterance graph, generated during the modeling phase, and a database of PDL_{SL} formulae, which is introduced in the next section.

5.1.3.1 Formulae database

The formulae database consists on a list of all the properties that an annotator wants to find on the video, encoded in PDL_{SL}. From the implementation point of view, the database is a simple text file, which is interpreted on-the-fly by an internal PDL_{SL} parser. As such, no specialized tools are needed for its creation, as a simple text editor is sufficient to generate the file.

Formulae in the database are assumed to be based on the same formalization primitives used by the Modeling module (see Subsection 5.1.2.1, p. 110). However, it is noteworthy to mention that this doesn't impede the database to contain formulae built upon other, undefined, primitives; this is a desirable characteristic in situations where future improvement is expected from the point of view of signal processing. For example, annotators may add formulae expressing changes in the Non-Manual Features (NMFs), expecting that in the future there might be a reliable way to recognize these changes automatically.

By relying on these modular phonetic descriptions, PDL_{SL} can be used to design formulae representing structures closer to the morphological level. For instance, consider the iconic production shown in Figure 5.20.

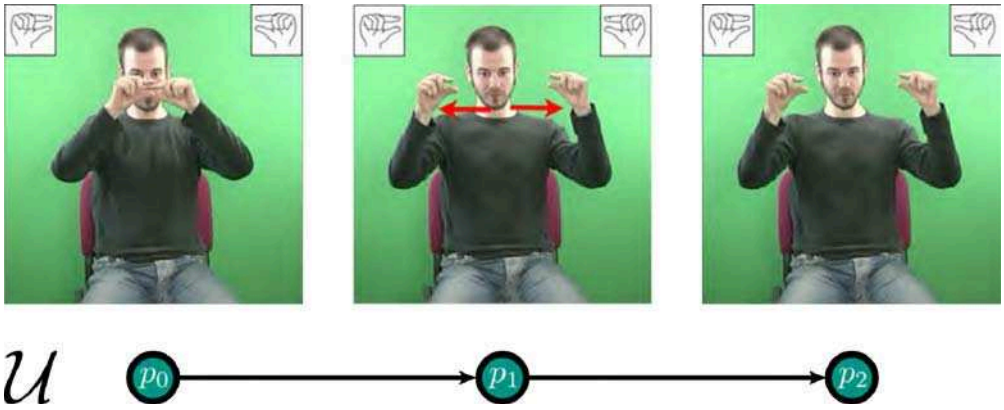


FIGURE 5.20: An iconic depiction of a thin and long object, and its corresponding utterance representation.

In the figure, a signer describes an undefined object, characterizing the object's thickness with his fingers and outlining its shape with hand movements. Roughly, the whole utterance conveys the idea of a thin straight edge, representing the qualities of the depicted entity: a thin long object. Even though the production is not a lexical sign, it roughly holds a HMH structure in the MHM model (see Chapter 2, Section 2.2.1.4, p. 29), where two static Holds (H) are separated by a simultaneous Movement (M) of the hands. Hence, PDL_{SL} can represent this interleaving with a formula of the type $H; \langle M \rangle H$, where each H is also a formula (describing static parameters), and the M is an action (capturing change parameters). The result is shown in Example 5.1.

Example 5.1 (`ICONIC` formula)**.**

$$(\mathbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L}) \wedge \mathbf{loc}(\mathfrak{R}, \texttt{HEAD}) \wedge \mathbf{cfg}(\mathfrak{R}, \texttt{CLAMP}) \wedge \mathbf{cfg}(\mathfrak{L}, \texttt{CLAMP}) \wedge \mathbf{touch}(\mathfrak{R}, \mathfrak{L}));$$
$$\langle \leftarrow_{\mathfrak{R}} \cap \rightarrow_{\mathfrak{L}} \rangle (\mathbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L}) \wedge \mathbf{cfg}(\mathfrak{R}, \texttt{CLAMP}) \wedge \mathbf{cfg}(\mathfrak{L}, \texttt{CLAMP}) \wedge \neg \mathbf{touch}(\mathfrak{R}, \mathfrak{L}))$$

■

The formula in Example 5.1, called `ICONIC` for simplicity, was built using the atoms presented by Definitions 5.7 and 5.8. In the top part, the formula establishes the information contained in the first H: the hands hold a certain configuration, they are positioned in front of the head and they are touching. The second line starts with an action denoting that both hands move, simultaneously, in opposite directions. The final section of the second line, introduces the result of the aforementioned movement over the first H: the hands remain in the same relative position, with the same configuration, but this time they are not touching. With respect to the latter, it is worth to mention that the atom $\neg\mathbf{touch}(\mathfrak{R}, \mathfrak{L})$ was explicitly asserted to avoid implementation ambiguities. Strictly speaking, the formula could have omitted the aforementioned atom and it would still be equivalent to the one presented.

The explicit declaration of the hands not touching, arises a peculiar characteristic of `ICONIC`: even though it is clear for a human, that separating both hands in opposite directions implies that they are no longer in contact, an automatic system will not necessarily have the ability to infer the same information. From here, it is easy to see that PDL_{SL} is capable of expressing inconsistencies — *e.g.* the hands separate and then touch — however this is not an issue, since it is not its role to avoid them. Rather, formulae consistency must be checked by the human designer upon creation or, ideally, tested automatically during verification.

Note that the presented formula is not unique: `ICONIC` could have been described differently, with different propositions. Also, observe that atoms of different articulators mix seamlessly within the formula, which is accepted by PDL_{SL} satisfaction rules (see Definition 4.20, p. 94).

Finally, formulae in the database may also correspond to either lexical signs or other simpler structures, as long as they are representable in PDL_{SL}. The idea is that, once the module knows about their existence, it can execute its verification algorithm regardless of the type of structure the formulae intend to describe. This results in a very flexible annotation creation process, since it is only necessary to adapt the database, and not the module itself, to produce different kinds of annotation. A more detailed description of the final verification process is explained in the next section.

5.1.3.2 Verification Algorithm

The principle of the verification algorithm, is based on the correspondence between formulae and utterance objects. For instance, let the formula from Example 5.1 be represented by `ICONIC`. It is the case that, in utterance object $\mathcal{U}$ from Figure 5.20, the

following equation holds:

$$\mathcal{U}, \overline{p_{02}} \models \texttt{ICONIC} \tag{5.1}$$

where $\overline{p_{02}} = p_0, p_1, p_2 \in \mathcal{U}$.

Therefore, by observing in which phoneme sequences each formula holds, it may be possible to determine whether the described property exists over a video segment or not. The complete verification algorithm is presented below.

Algorithm 5 Verifying PDL_{SL} formulae in utterances

```
1:  procedure UtteranceVerification(𝒰, 𝒟ℬ)
2:      Gloss ← {}
3:      for phoneme p_i ∈ 𝒰 do
4:          for formula φ ∈ 𝒟ℬ do
5:              find the largest j such that i ≤ j and
6:              if 𝒰, p_ij ⊨ φ then
7:                  Gloss.add((φ, p_ij))
8:              end if
9:          end for
10:     end for
11:     return Gloss
12: end procedure
```

Theorem 5.5 (Verification Complexity)**.** *For n interval changes and m formulae in $\mathcal{DB}$, the algorithm takes $\mathcal{O}(m \cdot n^2)$ time.*

Proof. Assuming line 7 takes $\mathcal{O}(1)$ time; the algorithm tests m formulae, in every phoneme of the $\mathcal{U}$. From Lemma 4.5, we know that $2n + 1$ phonemes are enough to capture all information in $\mathcal{SEQ}$, thus we have at most $\mathcal{O}(m \cdot 2n + 1)$. However, by line 5 we know that the algorithm tries to obtain the largest j such that $i \leq j$ at every state. Since utterances are capped to $2n + 1$ phonemes, it is safe to assume that the longest possible j decreases in 1 on each iteration. Thus, the total number of comparisons can be $\mathcal{O}(m \cdot \frac{(2n+1)(2n+2)}{2})$ on the worst case. Simplified, this formula gives us $\mathcal{O}(m \cdot (2n^2 + 3n + 1))$, which is bounded by $\mathcal{O}(m \cdot n^2)$. □

Algorithm 5 takes each phoneme $p_i \in \mathcal{U}$ and tests it against each formulae $\varphi \in \mathcal{DB}$, where $\mathcal{DB}$ denotes the PDL_{SL} database. For each p_i, it tries to find the longest sequence $\overline{p_{ij}}$ such that $\mathcal{U}, \overline{p_{ij}} \models \varphi$; this is, the longest video segment where a formula is satisfied (line 6). If such a sequence exists, it adds it to the proposed gloss, regardless if any other formula holds in the same sequence; the final proposal is not a unique annotation track, but rather a series of individual tracks (one for each formula). Figure 5.21 shows an example.

FIGURE 5.21: Example of a resulting gloss.

In Figure 5.21, two PDL$_{\mathsf{SL}}$ formulae describing the LSF signs "to recall" and "past", are satisfied in the utterance induced by the video segmentation; this is, equations $\mathcal{U}, \overline{p_{02}} \models \texttt{TO\_RECALL}_{\mathsf{LSF}}$ and $\mathcal{U}, \overline{p_{46}} \models \texttt{PAST}_{\mathsf{LSF}}$ hold. Note that Algorithm 5 returns an annotation track for each formula in $\mathcal{DB}$.

The module generates an editable file, that the user can modify directly by way of an annotation tool.

5.1.4 Annotation Tool

By thoroughly checking every state, the previously presented verification algorithm tries to maximize the number of possible hits; however, this process is usually prone to error, as several factors may mislead the module.

First of all, some of the existing phonemes may be co-articulations (see Chapter 2, Section 2.2.1.5, p. 31), which are indistinguishable from the others in the system. Since the algorithm is thorough, it will try to find hits in co-articulation phonemes, which may produce erroneous results. In addition, the noise introduced by tracking or segmentation errors may also induce the creation of incorrect phonemes, further complicating the problem. For this reason the final step of the process, shown by Figure 5.22, is simply the interaction with a human expert, who can modify the generated multi-track file by way of an annotation tool.

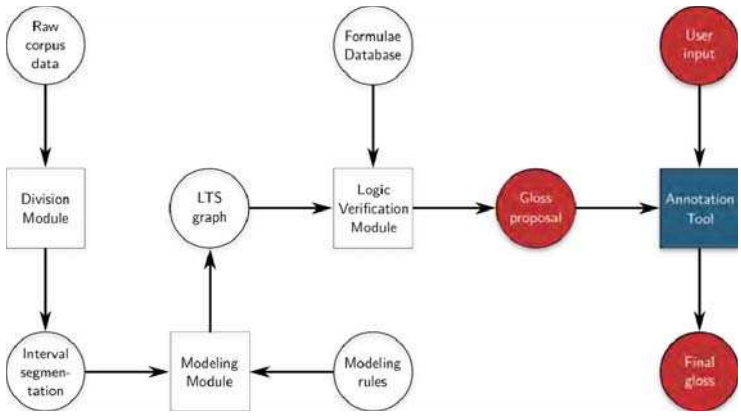


FIGURE 5.22: Block diagram of a generic PDL_{SL}-based SL lexical structure recognition system

In general, the implemented Modeling module generates a file for the ELAN annotation software. The idea is that the user can edit directly the resulting file in ELAN to obtain a final gloss, as depicted by Figure 5.23.

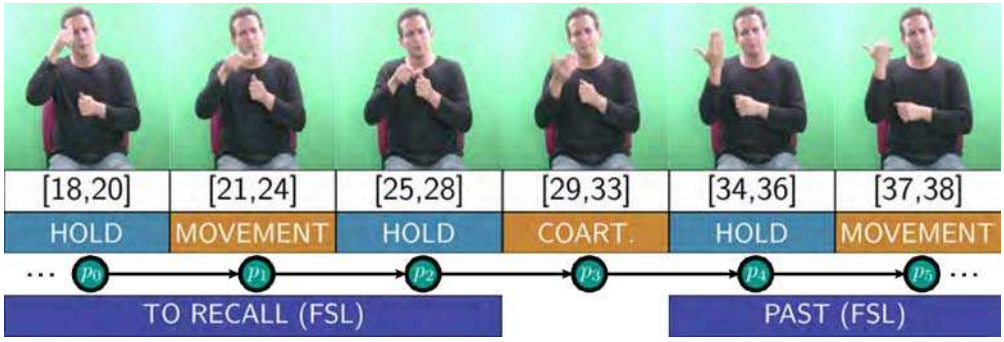


FIGURE 5.23: Example of a resulting gloss.

5.2 Dynamic Architecture Details

The framework introduced in Figure 5.1 (see p. 100), establishes the main development guidelines for any PDL_{SL}-based semi-automatic annotation system: one interval segmentation phase, one modeling phase and one verification phase. Furthermore, even though the modules presented in the last section were developed to work over a specific corpus, the underlying system architecture was designed to be a dynamic framework; this is, a system able to adapt on-the-fly to different corpora and analysis tools. As such, the system considers the situations where the modules implementing each phase need to be swapped: for example, when the tracking technology changes, or if new segmentation parameters are added. To this end, the system core implements the observer architecture shown in Figure 5.24.

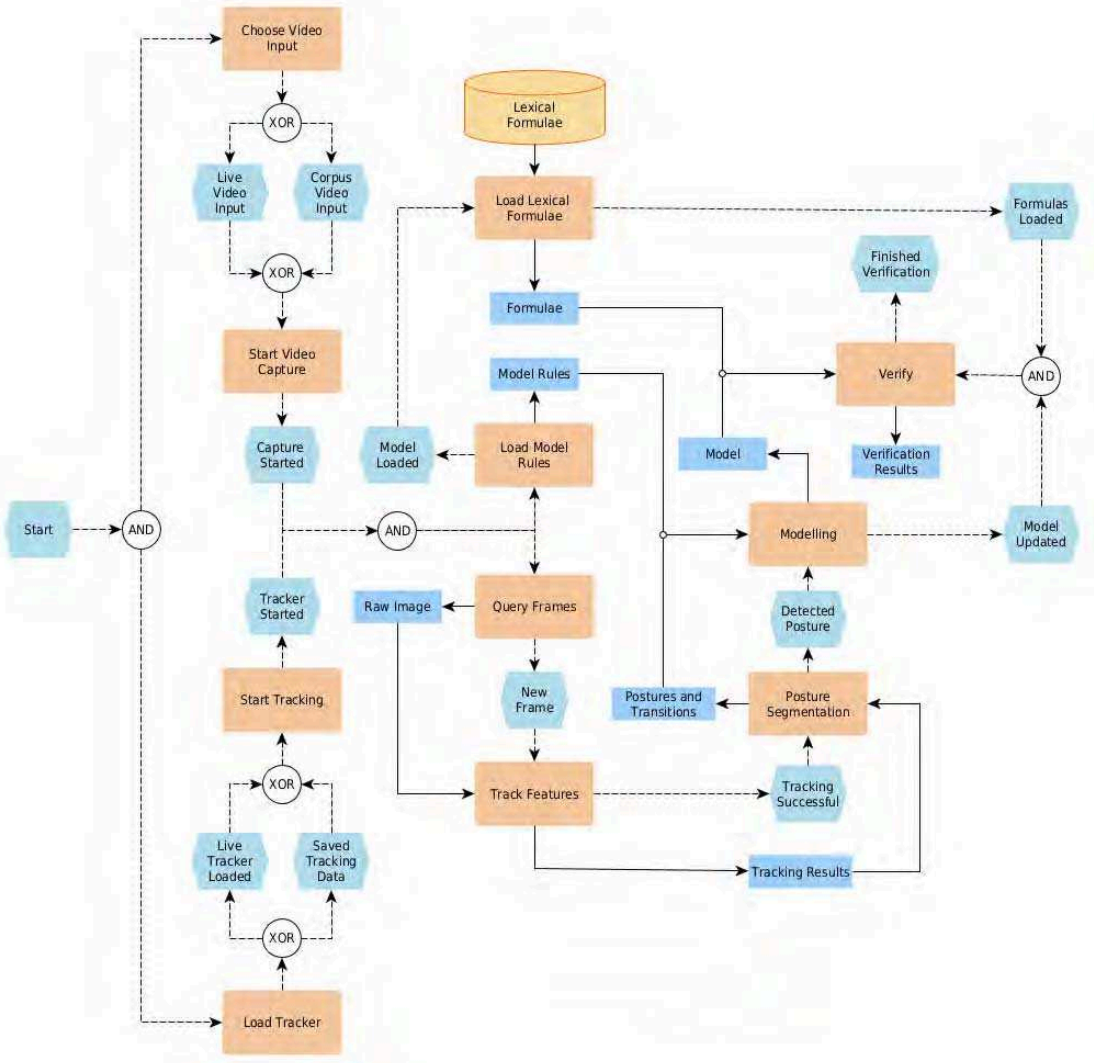


FIGURE 5.24: Information and control flow in the SL annotation framework

The main idea behind this design, is for the framework to be able to:

— use several tracking tools, depending on the characteristics of the input corpus;
— generate consistent utterance models from any combination of different trackers.

This task is not as direct as it may seem; recall that the Division and Modeling modules, presented earlier, are heavily dependent on the available tracking information. This implies that different combinations of tools may result in differing interval segmentations and, ultimately, utterance models. In addition, not every corpus can be analyzed with the same automatic tools. As such, the framework has to coordinate the different possible inputs, in order to ensure that every SL resource will be paired with compatible signal processing software. This process is entirely done by way of event-triggering: each time some significant action happens, in any part of the system, a system-wide announce is sent. This mechanism enables communication between modules, allowing them to act

appropriately depending on what events they receive. Furthermore, the architecture provides basic software interfaces for each of the framework's phases: roughly, it supplies implementation templates, detailing the inputs and outputs of each phase. With this, new modules developed to fulfill the tasks of any of the phases are forced to implement an interface, guaranteeing their compatibility. In turn, this promotes the creation of self-contained, interchangeable, modules, adapted to different research situations. For example, during the development phase, a preliminary Kinect-based Division Module was successfully hooked-up into the system, seamlessly introducing 3D movement information. No further development of this, or other 3D modules, was pursued; however, the successful substitution of one of the modules without breaking compatibility, goes on to show the advantages of adopting this kind of architecture.

The execution process of the architecture is fairly straightforward. At the beginning, a *Start* event is fired-up, prompting to load both a corpus resource and a processing tool. This task corresponds to the Division Module of the framework (see Subsection 5.1.1, p. 100), who loads a video and matches the selection with a compatible tracker. The loading process generates new events, indicating which selections were made, which can be used by the Modeling Module (see Subsection 5.1.2, p. 109) to select the appropriate modeling rules. Similarly, new events are generated so as to prompt the Logic Verification Module (see Subsection 5.1.3, p. 120) to select or discard database formulae, depending on which modeling rules are available.

Finally, even though the implementation details of the phases may change, the internal PDL_{SL} representation remains unchanged, as it is part of the system's core. In this regard, the verification process remains largely unchanged, as it is rather the contents of utterance objects themselves that may affected by the module swapping, and not the formalism's rules.

5.3 Chapter conclusions

In this chapter, a PDL_{SL}-based semi-automatic annotation framework was presented. Broadly, the framework's architecture is divided into three processing modules: the Division Module, the Modeling Module, and the Logic Verification Module. Together, they change a corpus resource into an abstract representation which can be semi-automatically annotated. More specifically, this chapter presented a concrete implementation of the system's modules, designed to analyze the Dicta-Sign corpus with a 2D head-hands position tracker.

Briefly, the process works as follows: the Division Module receives an untreated video from the corpus and obtains the hand tracking information. Afterwards, the module uses the obtained data to separate each articulation channel into an interleaving chain of static and change intervals, reflecting video segments where the articulator of interest either changes or not. To this end, the module analyses the dynamic characteristics of the hands. The resulting interval chains are passed to the Modeling Module, who uses a series of predefined modeling rules to transform the chains into a utterance object (roughly, a directed acyclic graph). Finally, the Logic Verification Module takes the generated utterance and uses it to search properties in it; each property corresponds to a PDL_{SL} formula, built by a human annotator. The final result can be edited by humans to correct the process as needed.

Even though the presented case is specific to Dicta-Sign, the underlying architecture is designed to enable the implementation of different versions of each of the modules, adapted to various types of corpora and tracking tools. This also helps to showcase the power of the formalism behind the implementation, as the chapter presented in detail how PDL_{SL} primitive sets were restrained to adapt to a real world scenario. In this regard, some insight was given into how the theoretical specification may differ from practical application, as purely mechanical problems (mostly related to the quality and format of the inputs) can easily affect the final results, depending on how each problem is solved.

The next chapter continues to delve into the contrast between theory and practice, by presenting some of the results obtained from using this implementation to search for PDL_{SL}-defined properties in video. In parallel, some of the formal language expressiveness issues will be addressed, alongside some discussion about how this can affect future computer-aided SL linguistic research.

Chapter 6

PDL_{SL} in Sign Language Research

This chapter presents some experimental results, obtained with the semi-automatic Sign Language (SL) annotation framework introduced in the previous chapter. During the experimentation phase, some technical adaptations to the original Propositional Dynamic Logic for Sign Language (PDL_{SL}) language were required, in order to ensure the *modularity* of formulae; this is, the formalism was extended to provide each formula with the capacity of being reused as part of some other formula. Also, some changes were required to account for undefined atoms: this is, propositions and actions used in the formulae database but not in the represented by the graph model. Finally, the chapter closes with some discussion about the expressiveness of PDL_{SL}, compared to existing transcription languages.

Broadly, the next section presents some technical modifications to the PDL_{SL} language, in an effort to simplify its use in linguistic research; although these are effectively theoretical, they result on, arguably, more comprehensible, reusable description formulae. Afterwards, the following section presents some experiments performed with the modified logic, as well as the obtained results. Finally, the chapter ends with a brief comparison between PDL_{SL} and other description formalisms, intended to provide an empirical insight on representativity issues, without lingering on stronger properties such as logic decidability.

6.1 Extending PDL_{SL} definitions

The original PDL_{SL} language, as presented in Chapter 4 (see Section 4.2, p. 83), enables experts to codify SL phoneme sequences by way of logic formulae; however, PDL_{SL}

provides no method to represent modularity. As a result, each time a PDL$_{\mathsf{SL}}$ formula is defined it has to be completely rewritten, even in cases when it shares multiple sub-formulae with other database members. Additionally, there are instances where it is useful to have the capacity of predetermining the truth value of a formula, by modifying atoms' valuation. This is useful in situations when an annotator is testing different hypotheses on a corpus, as it enables him or her to assume something is either true or false, and observe its effects; for instance, he or she may want to know what happens when every hand configuration atom is valuated as true, in systems where hand configurations are not recognizable. Regardless of the application, both cases can be solved by adding an *assignment* operator to the language (=), that can be used to name formulae and assign truth values. The extended syntax is presented in Definition 6.1.

Definition 6.1 (Extended PDL$_{\mathsf{SL}}$)**.** Let $\Lambda : \mathcal{E}_\varphi \rightarrow$ PDL$_{\mathsf{SL}}$ be a map taking $\mathcal{E}_\varphi$ to PDL$_{\mathsf{SL}}$ formulae.

$$\mathcal{E}_\varphi ::= \varphi \mid \neg\mathcal{E}_\varphi \mid \mathcal{E}_\varphi \wedge \mathcal{E}_\varphi \mid \mathcal{E}_\varphi = \top \mid \mathcal{E}_\varphi = \bot \mid \eta = \mathcal{E}_\varphi \mid \eta \mid \lambda\ \eta.(\mathcal{E}_\varphi)$$

where $\varphi \in$ PDL$_{\mathsf{SL}}$ and $\eta \in \mathcal{NAMES}$. ■

In addition to assignments, Definition 6.1 introduces *lambda expressions* [Barendsen 1994], to enable variable binding of atomic proposition and action parameters; as such, lambdas enable the description of groups of very similar formulae, for verification purposes. Note that the satisfaction rules for $\mathcal{E}_\varphi$ remain essentially unchanged: assignments are interpreted as PDL$_{\mathsf{SL}}$ formulae, as shown by Definition 6.2.

Definition 6.2 (Extended PDL$_{\mathsf{SL}}$semantics)**.** Recall the mapping Λ from Definition 6.1. The extended PDL$_{\mathsf{SL}}$ formulae are defined recursively by Λ as follows:

- $\Lambda(\varphi) \equiv \varphi$.
- $\Lambda(\neg\mathcal{E}_\varphi) \equiv \neg\Lambda(\mathcal{E}_\varphi)$.
- $\Lambda(\mathcal{E}_\varphi \wedge \mathcal{E}_\varphi) \equiv \Lambda(\mathcal{E}_\varphi) \wedge \Lambda(\mathcal{E}_\varphi)$.
- $\Lambda(\mathcal{E}_\varphi = \top) \equiv \Lambda(\mathcal{E}_\varphi)^\top$, denoting that every atom in $\Lambda(\mathcal{E}_\varphi)$ is substituted by $\top$.
- $\Lambda(\mathcal{E}_\varphi = \bot) \equiv \Lambda(\mathcal{E}_\varphi)^\bot$, denoting that every atom in $\Lambda(\mathcal{E}_\varphi)$ is substituted by $\bot$.
- for η $\mathcal{E}_\varphi$, if $\Lambda(\mathcal{E}_\varphi)$ is defined, then $\Lambda(\eta = \mathcal{E}_\varphi) \equiv \Lambda(\mathcal{E}_\varphi)$.
- if $\Lambda(\eta = \mathcal{E}_\varphi)$ is defined, then $\Lambda(\eta) \equiv \Lambda(\eta = \mathcal{E}_\varphi)$.
- For $\Lambda(\lambda\ \eta.(\mathcal{E}_\varphi))$, if $\Lambda(\mathcal{E}_\varphi) \in$PDL$_{\mathsf{SL}}$ is a formula where at least one of the atoms receives η as a parameter, denoted as $\Lambda(\mathcal{E}_\varphi)_\eta$, and every η parameter in $\Lambda(\mathcal{E}_\varphi)_\eta$ is defined over the same domain $\mathbb{D}$, then $\Lambda(\lambda\ \eta.(\mathcal{E}_\varphi))$ is equivalent to

$$\bigvee_{d \in \mathbb{D}} \Lambda(\mathcal{E}_\varphi)_d,$$

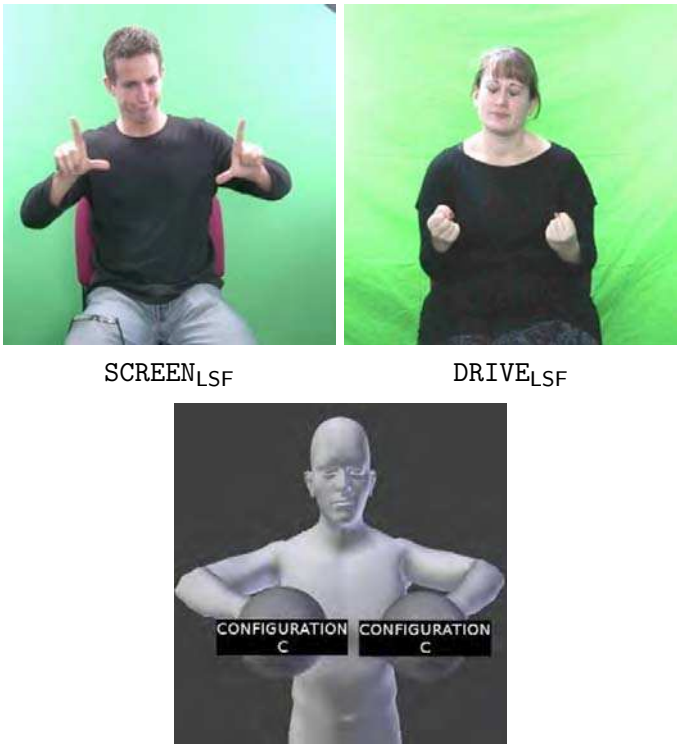
where $\Lambda(\mathcal{E}_\varphi)_d$ is the formula $\Lambda(\mathcal{E}_\varphi)_\eta$, with every occurrence of η substituted by a member of the parameter's domain. If a $\Lambda(\mathcal{E}_\varphi)_\eta$ doesn't exist but $\Lambda(\mathcal{E}_\varphi)$ is defined, then $\Lambda(\lambda\ \eta.(\mathcal{E}_\varphi)) \equiv \Lambda(\mathcal{E}_\varphi)$. Otherwise, it is not defined.

Finally, for some utterance $\mathcal{U} = (\mathcal{P}, \mathcal{T}_\mathcal{U}, [\![\cdot]\!]_\mathcal{U}, \mathcal{SEQ})$ satisfaction is given below:

$$\mathcal{U}, \overline{p_{ij}} \models \mathcal{E}_\varphi \iff \Lambda(\mathcal{E}_\varphi) \text{ is defined, and } \mathcal{U}, \overline{p_{ij}} \models \Lambda(\mathcal{E}_\varphi)$$

■

The presented extensions, improve the annotators' capacity to express succinct properties. For a simple example, observe the sign comparison shown by Figure 6.1.



SCREEN$_{\mathsf{LSF}}$ DRIVE$_{\mathsf{LSF}}$

COMMON STRUCTURE

FIGURE 6.1: Comparison of signs SCREEN$_{\mathsf{LSF}}$ and DRIVE$_{\mathsf{LSF}}$ sharing the same underlying structure. [1]

Figure 6.1 shows two French Sign Language (LSF) signs: SCREEN$_{\mathsf{LSF}}$ and DRIVE$_{\mathsf{LSF}}$. Both signs share the same underlying structure, characterized by both hands holding the same morphological configuration, while being spatially positioned opposite from one another. An annotator searching for every sign with the same underlying structure, where only hand configurations differ, can use a lambda expression to build a formula like the one presented in Example 6.1.

1. Corpus images obtained from Dicta-Sign project [Matthes 2012].

Example 6.1 (**opposition** lambda expression)**.**

$$\textbf{opposition} = \lambda c.(\textbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L}) \wedge \textbf{cfg}(\mathfrak{R}, c) \wedge \textbf{cfg}(\mathfrak{L}, c))$$

where the domain of c *is* $\mathcal{CFG}^{\mathfrak{R}} \cap \mathcal{CFG}^{\mathfrak{L}}$*.* ■

The formula can be modularized by way of the assignment operator, which enable the separation of **opposition** as depicted in Example 6.2.

Example 6.2 (modular **opposition** lambda expression)**.**

$$\mathit{hands\_config}(c) = \textbf{cfg}(\mathfrak{R}, c) \wedge \textbf{cfg}(\mathfrak{L}, c) \tag{6.1}$$

$$\textbf{opposition} = \lambda c.(\textbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L}) \wedge \mathit{hands\_config}(c)) \tag{6.2}$$

where the domain of c *is* $\mathcal{CFG}^{\mathfrak{R}} \cap \mathcal{CFG}^{\mathfrak{L}}$*.* ■

In Example 6.2, $\textbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L})$ indicates that $\mathfrak{R}$ lies in direction $\leftarrow$ with respect to $\mathfrak{L}$. Also, note that $\textbf{cfg}(\mathfrak{R}, c)$ and $\textbf{cfg}(\mathfrak{L}, c)$ denote that articulators $\mathfrak{R}$ and $\mathfrak{L}$, respectively, hold some configuration c; however, observe that, if they appear by themselves, these atoms are only defined if there exists a unique $c \in \mathcal{CFG}^{\mathfrak{R}} \cap \mathcal{CFG}^{\mathfrak{L}}$. By way of the lambda operator, c gains a different meaning; in formula 6.2, the expression implies that $c \equiv \mathcal{CFG}^{\mathfrak{R}} \cap \mathcal{CFG}^{\mathfrak{L}}$, thus indicating that opposition contains a sub-formula for every possible configuration common to both hands.

From the implementation point of view, lambdas can provide "syntactic sugar" to simplify formulae definitions. For example, by using **opposition** it may be possible to define signs SCREEN$_{\mathsf{LSF}}$ and DRIVE$_{\mathsf{LSF}}$ like shown in Example 6.3.

Example 6.3 (**opposition**-derived signs)**.**

$$\texttt{SCREEN}_{\mathsf{LSF}} = \textbf{opposition}(\texttt{L\_FORM})$$

$$\texttt{DRIVE}_{\mathsf{LSF}} = \textbf{opposition}(\texttt{FIST\_FORM})$$

where $\texttt{L\_FORM}, \texttt{FIST\_FORM} \in \mathcal{CFG}^{\mathfrak{R}} \cap \mathcal{CFG}^{\mathfrak{L}}$*.* ■

In the example, L_FORM denotes the morphological configuration of the hand where the thumb and the index fingers are held orthogonally (see Figure 6.1). Similarly, FIST_FORM denotes the configuration where the hand forms a fist. In general, **opposition**(L_FORM) and **opposition**(FIST_FORM) are two new formulae, interpreted by the system as being sub-formulae in **opposition** corresponding to configurations L_FORM and FIST_FORM, respectively.

Finally, assignments can help the system to better cope with situations where not every atom is present on an automatically generated utterance. Observe the formulae in Example 6.4, for the case where hand configurations are not recognizable.

Example 6.4 (**opposition** expression with a predefined valuation)**.**

$$hands\_config(c) = \lambda c.(\mathbf{cfg}(\mathfrak{R}, c) \wedge \mathbf{cfg}(\mathfrak{L}, c)) = \top \tag{6.3}$$

$$\mathbf{opposition} = \lambda c.(\mathbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L}) \wedge hands\_config(c)) \tag{6.4}$$

where the domain of c *is* $\mathcal{CFG}^{\mathfrak{R}} \cap \mathcal{CFG}^{\mathfrak{L}}$*.* ■

Since the value of hand configuration atoms is not calculated during the creation of the utterance object, it is implicit by PDL$_{SL}$ satisfaction rules that, regardless the value of $\mathbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L})$, **opposition** will always be false, since $hands\_config(c)$ is falsified by default. However, if the value of $hands\_config(c)$ is overridden to become true (formula 6.3), then satisfaction of **opposition** will depend on the value of $\mathbf{pos}(\mathfrak{R}, \leftarrow, \mathfrak{L})$, an outcome better adapted to the given situation. What's more, the definition of **opposition** itself didn't changed, as the override was done in one if the sub-formulae.

6.2 Experimental Results

By adapting the semi-automatic annotation framework (see Chapter 3, p. 41) to interpret the extended PDL$_{SL}$, it was possible to obtain some results over a set of simple PDL$_{SL}$ formulae. In particular, two specific cases were tested:

— Verification over purely static, or "True", Hold intervals: in which utterance segments are a strict interleaving of Holds and Movements (static and change phonemes).

— Verification over non-purely static Hold intervals: in which every utterance segment can either be interpreted as a Hold or as a Movement, depending on which articulators the formula refers to (static and change information is mixed in each phoneme).

The idea of this separation, was to observe how different interpretations of a utterance affected the final gloss. For example, consider the first case, where Holds are purely static phonemes. In this scenario, each "True" Hold corresponds to time intervals where no change was detected in any of the articulation channels. An example is shown in Figure 6.2.

In the figure, the only time intervals considered "True" Holds were the ones where no articulator was moving or changing in any way; intervals where one of the hands

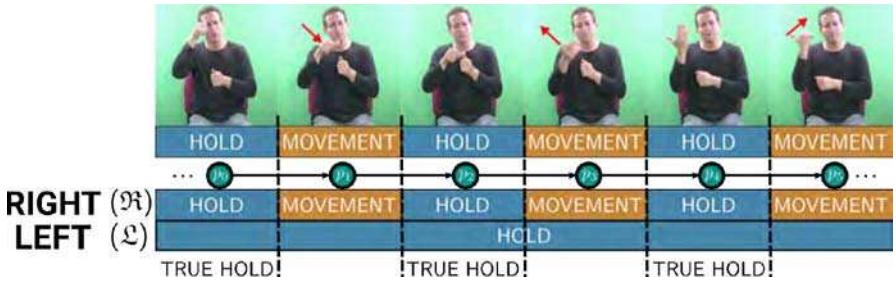


FIGURE 6.2: First Case: Considering only "True" Holds.

was moving and the other was static, were considered as Movements. The natural result of this classification, is that any interval between two purely static Holds will always be a Movement, inducing the strict interleaving previously mentioned. With this configuration, verification is prone to loose static data, since PDL$_{SL}$ formulae cannot be tested for satisfaction starting from any Movement interval — in accordance to the logic's semantics (see Chapter 3, Section 4.3, p. 94) — even though they may overlap with fixed articulators. For instance, in the figure, it is not possible to know anything about the left hand during any of the Movement intervals: the verification process can only test for static information on either a previous or subsequent "True" Hold, and it can only test for change atoms in the Movement itself.

It is important to note that the minimums subdivision presented previously (see Chapter 5, Subsection 5.1.1, p. 105) may induce Holds in this model. Recall from the text that the subdivision algorithm separates movement intervals with artificial Holds, which are inserted whenever it finds a speed minimum. A "True" Hold may appear if all articulators show simultaneous speed minimums, creating a static interval on every channel at that specific moment and, thus, being considered as a Hold globally. Nonetheless, even in this case, a strict interleaving of Holds and Movements is always respected, as implied by Theorems 5.2 and 5.3.

Contrary to the first testing case, the second doesn't display a strict interleaving of Holds and Movements: it considers that a Hold from the right hand can occur simultaneously with a Movement of the left, thus assuming that every phoneme can be of either type or a mixture of both. In this sense, even phonemes where all articulators change simultaneously — which in the first case would have been necessarily Movements — are considered as potential Holds and, as such, may be used to check satisfaction of static formulae. An example is shown in Figure 6.3.

In the figure, each utterance phoneme is considered as possibly having mixed information: this is, there is no global Hold-Movement sequence, just a chain of intervals where changes could exist or not. In this manner, every phoneme can be tested equally, and no

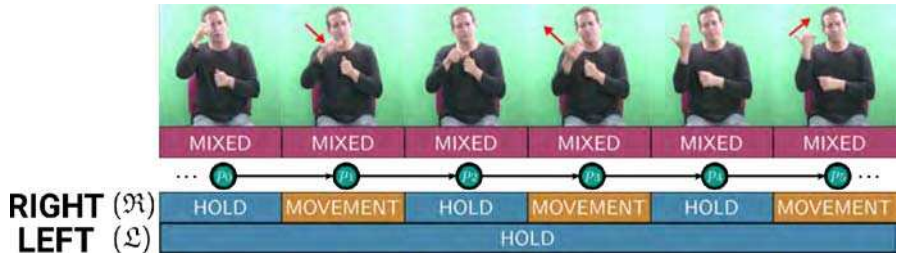


FIGURE 6.3: Second Case: Considering Holds and Movements simultaneously.

information whatsoever should be lost by the verification algorithm. Strictly speaking, individual articulation sequences remain the same as in the first case, the only difference being that the system avoids deciding globally whether an interval is of one type or other. As a result, the verification algorithm can decide on the fly if it considers a phoneme as a Movement or a Hold, depending on the formula being tested.

One way to think about the difference between the two cases, is that the first case is very coarse-grained, as it decides if a utterance phoneme is either a Hold or a Movement by looking at the state of every articulator, globally, during its associated time-interval; in contrast, the second case is more fine-grained, as it considers phonemes to be the aggregation of varying quantities of individual articulatory information, which can be used to change their type depending on formulae.

Either way, both cases were tested with the same set of formulae; a small PDL$_{SL}$ description database, consisting of the three descriptions presented below:

head anchor. $\lambda s.(\mathbf{loc}(s, \texttt{HEAD}); \langle moves \rangle (head\_active))$. Either of the hands is in the head region when some change occurs and, in the end, the head location remains active. Note that $\mathbf{skip}^a$ denotes any action in Π^a, as indicated by Definition 4.10 (p. 83). The sub-formulae are defined below:
— $moves = \mathbf{skip}^{\mathfrak{R}} \cup \mathbf{skip}^{\mathfrak{L}}$
— $head\_active = \mathbf{loc}(\mathfrak{R}, \texttt{HEAD}) \vee \mathbf{loc}(\mathfrak{L}, \texttt{HEAD})$

tap. $\lambda s.(\lambda w.(no\_touch(s, w); \langle moves \rangle (\mathbf{touch}(s, w); \langle moves \rangle (no\_touch(s, w)))))$. Either of the hands moves until they touch and, from there, either of them moves until they are not touching anymore. The *no_touch* sub-formula is defined as follows:
— $no\_touch(s, w) = \neg \mathbf{touch}(s, w)$

opposition. $\lambda c.(\lambda s.(\lambda w.(opposed\_hands(s, w) \wedge no\_touch(s, w) \wedge hands\_config(c))))$. Hands are opposite to each other, holding the same configuration. Note that $hands\_config(c)$ is the one defined in equation 6.3 and:
— $opposed\_hands(s, w) = \mathbf{pos}(s, \leftarrow, w)$

The formulae use lambda expressions to denote all possible combinations of hand-related atomic propositions. For instance, the formula for **tap** considers cases where either of the hands articulated the contact or separation.

The tests where performed over a single Dicta-Sign video with the following characteristics:

— the video had a duration of 11409 frames at 25 fps (7 minutes 36 seconds);
— it was manually annotated with 244 lexical glosses, where 12 were marked as "indecipherable" by the annotator;
— the task consisted of performing a LSF narration of a situation contrasting "Expectation vs. Reality".

To measure the *hit* ratio of the system, a human counted the number of apparitions in the video of the properties described above. Table 6.1 shows the observed quantities.

φ	**oppos.**	**tap**	**h. anch.**
Total	76	33	74

TABLE 6.1: Manually annotated apparitions of property formulae on one video.

The quantities of Table 6.1 were contrasted with the output of the Logic Verification Module (see Chapter 5, Subsection 5.1.3, p. 120) for each verification case. The results are analyzed in the next sections.

6.2.1 Verification over purely static Hold phonemes

For the case where only completely static time intervals are considered Holds, the execution of the verification algorithm reported the results shown in Table 6.2.

φ	**oppos.**	**tap**	**h. anch.**
Total	164	79	138

TABLE 6.2: Total reported hits over purely static phonemes on one video.

The table above, shows the total number of times each of the formulae were satisfied on the utterance representing the video. When manually comparing these results, with the human observations counted in Table 6.1, it was possible to generate the table presented in Figure 6.3.

In Figure 6.3, in order to determine the quantity of *hits* and *misses*, each reported gloss interval was compared with the human observations: if an overlap existed, it was counted as a hit. The human observations that weren't hit by the automatic gloss, were counted as misses. Finally, the last column of the table corresponds to *false positives*, *i.e* reported detections that didn't overlap with any human observation.

φ	Human Obs.		False+
	Hit	Miss	
opposition	67	9	97
tap	25	8	54
h. anchor	44	30	94

Table 6.3: Per-formula summary of the total number of observations found, missed and false positives for the first verification case.

Broadly, the results show a high recognition rate for **opposition** and **tap** (few instances were missed), but also a high quantity of false positives. Some of the erroneous **opposition** false positives were found in parts of the video where the signer had his arms either crossed, or resting over his knees; this is, even though the hands were correctly positioned to satisfy the formula, the video segment didn't depict signing. In other instances, it was observed that some of the false positives were due to tracking error or erroneous interval separation: for instance, in some cases Movements were the product of tracking noise which, when subdivided into minimums (see Algorithm 2, p. 105), induced the creation of "empty" segments (meaningless utterance states). This situation produced sequences of valid Holds, separated by sub-sequences of empty Holds and Movements; corresponding to changes induced by tracking noise.

Finally, during the algorithm's execution, a global articulation model (see Definition 4.13, p. 85) was inferred from the detected Holds and Movements. The result is shown in Figure 6.4.

The graph in Figure 6.4 displays a total amount of 205 states, representing all the static phonemes that were calculated during utterance creation. Overall, each state corresponds to a Hold phoneme that the signer depicted throughout the video sequence. Holds are connected between each other by 739 edges, representing all detected Movements in the video, produced by any of the articulators. From the annotation results, it is safe to assume that some of the depicted Movements were induced by tracking noise. Similarly, some of the Holds necessarily correspond to meaningless postures (like when the signer crossed his arms), or may be the product of small changes induced either by noise or by the choice of segmentation thresholds. All the same, the diagram goes on to show that, globally, the number of complex interactions is very high, thus producing a large number of empty Holds. In this view, verification based on non-purely static Hold phonemes may represent an advantage; in theory, it may provide a way to avoid verification over the empty Holds, based on ignoring information that could had been induced by tracking or thresholding noise.

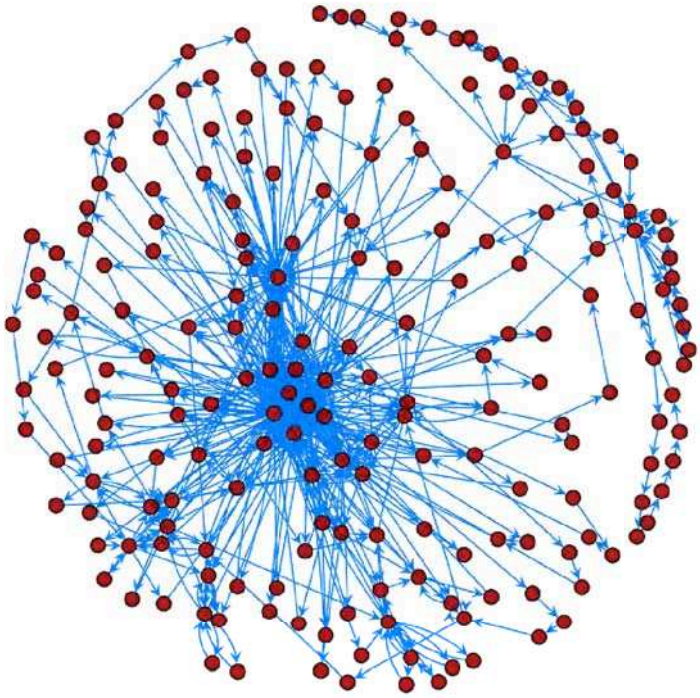


FIGURE 6.4: Depiction of the inferred articulation model from one video.

6.2.2 Verification over non-purely static Hold phonemes

The verification results for the case where any phoneme in the utterance can be considered a Hold, are reported in Table 6.4.

φ	**oppos.**	**tap**	**h. anch.**
Total	390	48	84

TABLE 6.4: Total reported hits over mixed phonemes on one video.

The most salient characteristic of these results with respect to the previous case, comes from the quantity of glosses that the algorithm produced. The quantities in Table 6.4 show that, as the volume of possible Holds increased, the number of reported hits for **opposition** exploded. This was to be expected, since **opposition** is verified in Holds alone: in this case, as any phoneme can be considered a Hold, the quantity of testable phonemes increased, explaining the larger number false positives. On the other hand, for modal formulae, the overall number of reported positives decreased. In either case, the overall quantity of good hits remained stable, as shown by Table 6.5.

The drastic reduction of false positives in formulae with modal properties may seem counter-intuitive at first glance. However, a deeper analysis revealed that the algorithm effectively managed to discriminate and ignore some of the empty Holds during the verification process, something that was not possible on the first verification case.

φ	Human Obs.		False+
	Hit	Miss	
opposition	71	5	314
tap	31	2	15
h. anchor	38	36	10

Table 6.5: Per-formula summary of the total number of observations found, missed and false positives for the second verification case.

For example, suppose that there are two Holds h_0, h_1, connected by a movement m_0, such that: $\mathcal{U}, h_0, m_0, h_1 \models$ **head anchor**. More precisely, suppose that $\mathcal{U}, h_0 \models \mathbf{loc}(\mathfrak{R}, \texttt{HEAD})$ and that $\mathcal{U}, h_0, m_0, h_1 \models \langle \mathbf{skip}^{\mathfrak{R}} \rangle (\mathbf{loc}(\mathfrak{R}, \texttt{HEAD}))$, meaning that the right hand was in the head, moved, and ended up in the head again. If the left hand introduces tracking noise (n_0) between the two Holds then the sequence gets subdivided either as h_0, n_0, h'_0, m_0, h_1 or h_0, m_0, h'_0, n_0, h_1. For the first articulation case, h'_0 is an empty state where $\mathcal{U}, h'_0 \models \mathbf{loc}(\mathfrak{R}, \texttt{HEAD})$, thus either $\mathcal{U}, h'_0, m_0, h_1 \models$ **head anchor** or $\mathcal{U}, h_0, n_0, h'_0 \models$ **head anchor** are valid. If more noise is introduced between Holds, more false positives are detected. However, on the second case, the same scenario would render a single result: by the PDL$_{\mathsf{SL}}$ semantics, the formulae would be verified over the articulation sequence of the right hand, where no noise was introduced. Moreover, all h_0, n_0, m_1 and h_1 are mapped to a single transition in the articulation sequence of $\mathfrak{R}$ (see Definition 4.18, p. 4.18), thus guaranteeing that every gloss produced from any of those phonemes is the same; hence, the quantity of false positives is reduced. The comparison of the results from both cases, is summarized in Table 6.6.

φ	Human Obs.	1st Case			2nd Case		
		Hit	Miss	False+	Hit	Miss	False+
opposition	76	67	9	97	71	5	314
tap	33	25	8	54	31	2	15
h. anchor	74	44	30	94	38	36	10

Table 6.6: Global per-formula summary of the total number of the obtained results.

The table shows that, in terms of overall hits, there is no real advantage from one case to the other; in fact, it suggests that some formulae could benefit from being verified over purely static Hold sequences, whereas some others would be better off in non-purely static sequences. However, due to the small size of the testing samples, it is not possible to provide strong claims on the issue without further testing both over larger corpora and with more pertinent databases. Regarding the latter, in the next section some final reflections are given about PDL$_{\mathsf{SL}}$ expressiveness, and on how property databases can be created automatically from existing SL resources.

6.3 PDL$_{SL}$ and other SL notations

The simple database presented in the previous section, the one used to test the semi-automatic verification framework, was created manually; a human directly wrote PDL$_{SL}$ formulae from corpus observations. However, for the most part, this is not a straightforward task. For users unfamiliar with formal logics, building formulae like the ones proposed can be an issue, as they require a certain level of familiarity and experience. In addition, as presented in previous chapters, several SL representation projects have been developed over the years, permitting the creation of several annotation resources that are still available to this day. For these reason, it might be important to establish a correspondence between PDL$_{SL}$ and other existing representation formalisms as this could:

— enable familiarization of the users towards PDL$_{SL}$; and,
— provide information on how to convert one formalism into the others, permitting the automatic creation of PDL$_{SL}$ databases from existing resources.

To this end, in this the next paragraphs some comparisons between existing representation languages are given, providing a little insight on how they relate to PDL$_{SL}$. However, the reader should know that the information provided by this section is by no means thorough; it is not intended to give construction recipes, but rather to measure informally how such a recipe may be created for different formalisms.

6.3.1 Synthesis languages

As reviewed in Chapter 3, the domain of SL synthesis usually requires sign representation formalisms (see Subsection 3.2.1, p. 51), in order to indicate their systems how to execute a sign. As mentioned in the aforementioned Chapter, these notations tend to be very geometric-oriented, as they have to be fairly specific about movement and articulator positioning. Because of this, some especial considerations have to be taken in account if one is to transform such descriptions into PDL$_{SL}$ formulae. In this subsection, some of these considerations are addressed by analyzing one of these notations in particular: ZeBeDee.

6.3.1.1 ZeBeDee

ZeBeDee is a SL representation notation, oriented towards language synthesis (see Chapter 3, Subsection 3.2.1.2, p. 54). It was developed to describe SL lexical units by way of geometric constraints and *contextual dependencies*: roughly, the language

permits the representation of spatiotemporal relationships — between articulators or with space locations — instead of parametric characterizations, resulting in highly reusable descriptions that are neither over-specified nor under-specified. The temporal context is established by way of time-interval sequences, inspired by the phonetic model by [Johnson 2011]. It is usually used in conjunction with AZee, which provides stronger expressive capabilities to the language, enabling higher level descriptions (see Chapter 3, Subsection 3.2.1.3, p. 54).

Space in ZeBeDee is of the utmost importance. As such, the formalism defines a three dimensional space around the signer, where geometrical descriptions can be provided in terms of points and vectors. As such, it is by way of spatial locations that an annotator can specify articulation places, not only on the space around the signer but also to name corporal regions; for this, ZeBeDee provides the operation `Place @P at` K, which assigns a name of articulation `@P` to a geometrical object K.

In contrast, PDL$_{\mathsf{SL}}$ defines space coordinates implicitly, as they are never specifically named in the language, but rather by the defined primitives. PDL$_{\mathsf{SL}}$ also establishes a three dimensional coordinate space (Ψ_{SL}), but it makes a clear separation between its members and their meaning, limiting the introduction of the latter to formulae. As an example, consider the set $\mathcal{LOC}^a$, which identifies locations for an articulator a as members of $\mathcal{NAMES}$ and not coordinates; from Definition 4.4, the link between names and location coordinates is given by the function $\mathcal{C} : 2^{\Psi_{\mathsf{SL}}} \rightarrow \mathcal{NAMES}$, which is defined for a name in $\mathcal{NAMES}$ only if there is a coordinate associated to it. With this in mind, PDL$_{\mathsf{SL}}$ can interpret `Place @P at` K as "define `@P` in $\mathcal{C}$ such that $\mathcal{C}(@\mathtt{P}) = K$ and $K \in \Psi_{\mathsf{SL}}$".

With respect to the name of corporal places, ZeBeDee takes as main reference both body sites and parts of the signer's skeleton. The first ones, surface body locations that the signer can touch, are denoted by the symbol `@`, and provide regions such as the sternum (`@ST`) and the like. In PDL$_{\mathsf{SL}}$, this abstraction corresponds directly to the members of the set $\mathcal{BODY}$, and thus `@St` can be read as "`ST` $\in \mathcal{BODY}$".

For the skeleton, ZeBeDee names every relevant bone (*segment*) individually, from the points of the fingers to the torso. In addition, the language provides a nomenclature for the right (`r`), left (`r`), dominant (`s`) and non-dominant (`w`) sides: for example, the dominant arm can be denoted as `!arm(s)`, where the character "!" indicates that `arm` is part of the skeleton. Note that `!arm(s)` can be the same segment as `!arm(r)`, depending how the signer is considered. In addition, ZeBeDee links body sites with skeleton segments; roughly, sites correspond to the surface of some segment.

PDL_{SL} do not give a direct representation for ZeBeDee's skeleton as is, specially because ZeBeDee does a semantic separation between the skeleton and the surface; however, it does provide the basic notion, by way of the set $\mathcal{ART}$. Recall from Definition 4.4 that the set $\mathcal{LOC}$ is created by pairing articulators in $\mathcal{ART}$ to both named geometrical locations and members of $\mathcal{BODY}$ (even though $\mathcal{ART} \subset \mathcal{BODY}$). Then again, these pairings are not defined by the language but by the designer, therefore as long as the skeleton parts are in $\mathcal{ART}$, the semantic separation can be implicitly established by $\mathcal{LOC}$.

ZeBeDee uses the previously explained definitions to create Holds and Movements (for the authors *key postures* and *transitions*). In particular, Holds differentiate from Movements by the nature of their respective parametric informations. For PDL_{SL}, this corresponds to the members of the sets Π and Φ; recall that Π contains information about change (Movements) and Φ about static postures (Holds). In this regard, for both formalisms would seem to be compatible, since both treat statements as predicates; Chapter 5 provides more information about PDL_{SL} in this regard. Nonetheless, as what happens with locations, PDL_{SL} separates the semantics of the predicate from itself, and thus relies on the implementation of modeling rules (see Definition 5.9, p. 117) to provide meaning. For this reason, ultimately any symbol in either atom set can represent any ZeBeDee statement, as long as there exist a modeling rule defined in terms of $\mathcal{ART}$, $\mathcal{BODY}$, $\mathcal{LOC}$ over the coordinate space Ψ_{SL}.

Regarding temporal structure, by design ZeBeDee can express sequences of Holds and Movements by grouping parameters into two structures: Keypostures and Transitions, respectively. These are built upon temporal notions such as time duration, which is not directly built into PDL_{SL}; the language is capable of expressing sequences (see Definition 4.12, p. 84), but it doesn't consider duration. Recall that, to model corpora, it was required to define time-stamping functions for articulation sequences (see Definition 4.15, p. 87). Because of this, in order to introduce time to a formula, it is necessary to define *duration* as a statement, expressible in terms of atoms: an atom **duration** can be added to the sets Φ and Π, so as to assert inside formulae how much time a phoneme lasts. However, the reader should be aware that this entails the creation of modeling rules to define the meaning of time, just as it was done with space (see Definition 4.3, p. 80).

Finally, ZeBeDee provides a series of auxiliary tools like macros and loops, geared towards improving the experience of using the language rather than SL itself. There is no clear way to establish correspondence between these additional structures and PDL_{SL}; nonetheless, as long as their final results are expressible in ZeBeDee's primitive types, it should be possible to encode the same information as PDL_{SL} formulae.

To sum things up, a ZeBeDee description can be converted into a PDL$_{SL}$ formula by following the pattern presented below:

1. for each Keyposture, transform the contained information into a non-modal PDL$_{SL}$ formula;
2. for each Transition, transform the contained information into a $\mathcal{A}_{SL}$ action formula (see Definition 4.11, p, 84);
3. combine the results of this transformations into a modal PDL$_{SL}$ formula, respecting the sequence defined by the ZeBeDee description.

From the previous paragraphs, it has been argued that there is a straightforward correspondence between PDL$_{SL}$ terms, and the information described by Keypostures and Transitions. In addition, ZeBeDee is already well adapted to be parsed by way of a computer system: accordingly, the entire transformation process roughly consists of "unfolding" ZeBeDee descriptions into PDL$_{SL}$ formulae, which can be done automatically. The situation is slightly different of what would happen with SL transcription notations, as these are less computer-friendly than ZeBeDee, and define parameters in a more abstract manner, as explained in the next section.

6.3.2 Transcription languages

More directly related to linguistic research, other representation notations are less concerned on geometrical accuracy, but rather on how to avoid information loss in SL transcription (see Chapter 3, Section 3.1, p. 42). As such, these transcription-oriented notations tend to be very symbolic, and rely on human interpretation to decode their information. Here, one notation in particular is compared to PDL$_{SL}$: the Hamburg Notation System (HamNoSys).

6.3.2.1 HamNoSys

HamNoSys is a transcription notation developed to represent SL resources in written form (see Chapter 3, Subsection 3.1.2 p. 44). In general, HamNoSys captures signs' features by providing an initial parametric posture and a sequence of actions affecting said posture. To this end, the language provides a series of symbols representing places of articulation, hand configurations, orientation and Non-Manual Features (NMFs).

PDL$_{SL}$ and HamNoSys share the same basic principle; just as HamNoSys, the logic permits establishing a departure posture and denoting modifications over it. However, unlike HamNoSys, modal PDL$_{SL}$ formulae explicitly express how the resulting posture

should look $(posture; [action](result))$, whereas HamNoSys actions' results are left implicit. Mainly, this is because the semantics of HamNoSys symbols are broader than those defined in PDL_{SL}: they contain more information than a PDL_{SL} atom.

For example, observe the action $\downarrow^{\mathfrak{R}} \in \Pi^{\mathfrak{R}}$, defined for the study of 2D corpora. The meaning of the action is linked to a phoneme in a utterance graph (see Definition 4.17), where the y position of the right hand decreases monotonically while the x position remains unchanged. Further information about the right hand has to be asserted elsewhere, with other atoms. In contrast, the HamNoSys symbol ↓ implicitly accounts for the hand's location, position and, if a concatenation of symbols exist, even the path it follows.

As a result, in order to be able to transform HamNoSys strings into PDL_{SL} formulae, a mechanism to calculate postures by the successive application of actions is required. This hints to the development of a PDL_{SL} calculus, based on morphological rules about the movement of the human body. Otherwise, the alternative is to delve into the creation of thorough articulation models in the spirit of the one presented on Figure 6.4; in this case, the idea would be to interpret the initial HamNoSys posture, find it in a graph representing an articulation model and, from there, determine the resulting state by the application of the specified actions. Once the end state is found, the resulting posture can be encoded as a PDL_{SL} formula.

With respect to the remaining symbols, the transformation is pretty straightforward. As with ZeBeDee, HamNoSys defines a coordinate system akin to Ψ_{SL} in PDL_{SL}. From here, the interpretation of language statements such as orientation or location, can be defined by way of the PDL_{SL} primitive mappings and model rules. Finally, the definition of the members of sets $\mathcal{BODY}$, $\mathcal{ART}$ and $\mathcal{CFG}$ is less strict than with ZeBeDee, permitting the use of broader labels for their definition; as a matter of fact, the contents of these sets could even be denoted directly by HamNoSys symbols, as they are considered in $\mathcal{NAMES}$ by definition. In this regard, the only real issue comes from calculating the expected effects of actions which, as indicated, is usually a task performed by humans.

6.4 Chapter conclusion

In this chapter, an extension to the PDL_{SL} was presented, providing a way to work with modular formulae during semi-automatic annotation tasks. The extension was used to build a simple PDL_{SL} database which, in turn, was used to generate some practical results over the annotation framework from Chapter 5 (p. 99). In the final section, a brief analysis comparing the expressiveness of PDL_{SL} against two different SL written

notations is given; the main objective, is to show how existing SL resources could be turned into PDL_{SL} representations, simplifying the work of building formulae databases.

Broadly, the results presented in this chapter highlight the possible pitfalls that PDL_{SL} may encounter when applied to semi-automatic annotation. First of all, the formal language had to be extended for it to adapt to the chaotic conditions that formulae might find in practice (namely, lack of data). This was paired with a comparison of how two varying interpretations of formulae can render different results in practice, as showcased by the two verification cases in Section 6.2. The insights gained from the examples, have helped shaping the semantics of the formal language, and opened future research possibilities based on the obtained results.

Chapter 7

Conclusions and Perspectives

This present work has introduced a formal representation for Sign Language (SL) phonology, oriented towards the development of semi-automatic annotation tasks. In particular, the proposed formalism was developed to help solving some of the common problems existing in computer-based SL research, notably:

— the lack of a standard written representation for SL corpora, which forces automatic systems to work directly with raw data (such as video images);
— the heterogeneity of corpus-data, which impedes the creation of large, uniform, language information repositories, thus curbing the effectiveness of machine-learning algorithms and the like; and,
— the difficulty to systematically annotate video information, taking into account the variability of annotation tasks and the limited availability of SL experts to perform and correct them.

In the next paragraphs, a brief analysis is given on how the Propositional Dynamic Logic for Sign Language (PDL_{SL}), the proposed representation formalism, contributes to solve the above issues, alongside some perspectives of future work.

7.1 On SL corpus representation

Regarding SL representation, PDL_{SL} provides two basic abstractions to encode SL information: logic formulae (to represent SL phonetic-phonological interactions) and graph-based utterances (to symbolize corpora). Both of them are closely related to previously existing transcription efforts, with the difference that they center on flexibility: they permit the creation of new symbols as a core feature of the language, so as to adapt to any use case in research.

With respect to corpus representation, PDL_{SL} contributes to reduce heterogeneity by permitting the transformation of SL corpora into a well-defined graph utterances, as shown in Chapter 5 (see p. 109). Once the aforementioned transformation is done, further analysis can be executed directly over the resulting graph, without resorting to further processing over the original data; this means that if several corpora, with different technologies, are represented under the same PDL_{SL} primitives, then any algorithm designed to analyze one of them will also work on the others. As a result, the process can potentially increase the quantity of exploitable data, regardless of the origin of each corpus.

However, there are heterogeneity-related limitations to this process. For instance, the process presented in Chapter 5, showcasing the formalization of a 2D corpus, still relies on specialized tools to generate the graph (in this case, a two-dimensional hand position tracker). If the corpus would have had motion capture data, the same process would have been performed by a different algorithm, adapted to interpret information as needed. Despite this, when applied to recognition tasks, PDL_{SL} presents an advantage with respect to other methods: heterogeneity is only an issue *before* any linguistic analysis takes place. In contrast, existing sign recognition efforts have to deal with the problem *during* the analysis (see Chapter 3, p. 56), limiting the application of their methods to the specific corpus for which they were created.

The same situation goes to shows how PDL_{SL} contributes to avoid a common problem, linked to the lack of a standard written representation: recognition algorithms don't need to work on raw data, since utterance objects already represent the information contained in the video. In addition, these graphs are more expressive than transcription languages (see 6 p. 140), as they are unbounded on the quantity of information they contain.

On the other hand, unbounded information may also prove to be a disadvantage in some cases; as explained in 6, the granularity of information may affect the results of verification in recognition tasks, resulting in either too many false positives or no hits at all. These problems, called *underspecification* and *overspecification* in classical formal methods [Bergstra 1991], oblige annotators to develop *adequate* rules for their use case, which is not always evident and may be subject to trial and error.

7.1.1 Perspectives

From a theoretical standpoint, the most natural path for future PDL_{SL} research leads towards the analysis of articulation models; this is, the underlying structures used to produce SL utterances, as defined in Chapter 4 (see Definition 4.13 p. 85). Contrary to utterance objects, which represent only specific instances of language use, articulation

models hold generalizations about the SL of interest, which makes them useful in contexts beyond annotation.

Broadly, articulation models are interesting in the sense that they may contain the basic generation rules for any SL production. Recall from Chapter 2 that SLs from different regions tend to be unique; this implies that each SL has different sets of valid parametric postures, arranged by distinct combination patterns. This would indicate that, together, all the articulation models of a single SL should be characteristic of said language or, at least, of its linguistic family; however, PDL_{SL} doesn't yet define a way to unify these models consistently. For this reason, it may be worthy to explore how various articulation models fit together into a single structure, as it has potential applications in areas such as SL synthesis or SL recognition. For example, it may provide more accurate generation models (useful in synthesis), establish proximity measures between different SLs (useful for low-resource Natural Language Processing (NLP)) and help in the identification of patterns pertaining to linguistic phenomena (useful for recognition). They may also help to develop a formal calculus for PDL_{SL}, grounded on the specific properties derived from a global articulation model.

Additionally, the current state of PDL_{SL} provides a better starting point than before for the exploration of articulation models: the implementation of the semi-automatic annotation framework, has shown the potential that PDL_{SL}-based systems have to infer articulation models from data (see Figure 6.4, p. 138), opening the door to new measures akin to those used to calculate phoneme frequencies in vocal languages.

7.2 On PDL_{SL}-based semi-automatic annotation

From the point of view of corpus annotation, Chapter 5 presented the implementation of a semi-automatic annotation framework, based on the principles introduced by the PDL_{SL}. The proposed system, provides a general architecture, representing the entire process from transforming a raw corpus video into an abstract representation, to the generation of a gloss proposal that can be edited by a human expert. In particular, the described system was developed to annotate Dicta-Sign corpus, using not only video data but also the results of a 2D head-hand tracker.

Notably, the annotation process is based on a database of PDL_{SL} formulae, describing which properties are to be searched for in the graph representation of each corpus video. The proposed algorithm, enables the automation of the annotation process, since the same database can be used seamlessly to treat different corpus videos, as long as their graph representation remains consistent. In this way, annotators can simplify the overall

process by defining a single database that they can use repeatedly for each new video. Still, this process also shows some limitations.

The principal limitation, comes from the formulae contained in the database: they are affected by similar overspecification issues like the ones mentioned in the previous section. Recall from Chapter 4 that PDL_{SL} formulae are designed to be broad; as a result, PDL_{SL} establishes no standards for annotation, leaving the work in the hands of human annotators. The complication comes from the fact that each expert has the liberty to describe properties following his or her own standards, which may greatly differ from any others. This means that, theoretically, there could be at least as many individual annotation templates as there are SL experts. In turn, this situation could end up mimicking the already existing problematics in annotation, since the resulting databases would only be usable on their original context, making difficult their reutilization over different corpora.

Also, even if annotation formulae were uniform across corpora, this wouldn't necessarily guarantee that a database would always be reusable without hiccups; recall that the modeling rules used to create graph representations from videos (see Chapter 5, Subsection 5.1.2, p. 109), depend on having the necessary tracking or recognition tools to determine the truth values of atoms in a corpus resource. This means that not necessarily every atom used in a formula will be detectable in every corpus, nor that they will always be correctly valued (tracking tools are prone to error, video conditions could be suboptimal, etc); both situations could influence the accuracy of the resulting annotation, regardless of the quality of the database.

Finally, the given implementation displays less-than-optimal algorithmic complexity; for the time being this is barely an issue, as the number of articulators is small enough to not represent a serious performance hit. However, the obtained values show that combinatorial explosion could rapidly become unmanageable without much effort, which could hinder the system's scalability.

7.2.1 Perspectives

The most pressing issue regarding the semi-automatic annotation system, is for broader capabilities to be developed into each of the existing modules, so as to try the process in other corpora. Crucially, the integration of modeling rules for the three-dimensional case is becoming a necessity, in order to widen the scope of the system; as presented in Chapter 3, various corpora are recorded with the help of motion capture devices or 3D equipment. However, this poses a number of challenges to automatic corpus analysis, starting with the interval segmentation from the Division Module.

Recall from Chapter 5 (p. 104), that the Division Module selects video sequences based solely on tracking information. Thus, it is safe to assume that the introduction of an additional spatial dimension would, for better or worse, modify the results calculated by the module. The same case can be made for any additional tracking or recognition information like hand-configuration, Non-Manual Features (NMFs) or body posture: the basis of the segmentation is the identification of changes, and new data may affect what the module considers a change. In brief, this means that any segmentation based in raw data requires a careful examination of the possible outcomes, suggesting the need for new linguistic research regarding phoneme identification under varying conditions.

Similarly, adapting the framework to new corpora entails a reworking of the modeling and verification algorithms, in terms of optimization; remember that the calculated complexity in the Modeling (see 5.4, p. 118) and Verification (see Theorem 5.5, p. 123) modules, is by no measure small. As such, the addition of new information prone to introduce change intervals, can rapidly become unsustainable without any optimization.

On a final note, a better integration of the framework with some manual annotation mechanism is desired, as it may improve the process robustness; that is, instead of providing an output file to be edited elsewhere, it would be better for users to have direct access to the annotation in real time, to enhance the verification process as it happens. This may also help reduce the verification algorithm's complexity, which could benefit from external data to avoid systematically testing every possible state. Likewise, introducing on-the-fly corrections may also enable the use of machine learning algorithms to learn from these, dynamically improving both the modeling and verification processes.

7.3 On the empirical application of $\mathrm{PDL}_{\mathsf{SL}}$ in linguistics

Finally, something has to be said with respect to the multidisciplinary nature of this work. The presented formalism is, first and foremost, a tool. As such, the context of its intended use is to be taken in account for its development. Recall from Chapter 3, that annotation tasks in linguistics are very variable, that they respond to different research needs and they are, ideally, in constant iteration. In this regard, $\mathrm{PDL}_{\mathsf{SL}}$ formulae have properties that can enrich the corpus annotation process, such as modularity; this is, the ability to define formulas as groups of reusable blocks, as presented in Chapter 6.

In practice, modularity enables the rapid development of annotation iterations; by changing the definition of a single module, an annotator may also change every formula built with the same module, thus potentially accelerating the subsequent production of new annotation batches. This also enables the constant testing of linguistic hypotheses

by way of PDL$_{\mathsf{SL}}$ databases, as it becomes relatively simple to override large groups of formulae, execute the verification algorithm and compare results between executions.

Despite these advantages, cases like the ones mentioned above assume familiarity with formal tools, a domain which is not necessarily well-known in linguistics. Moreover, the complexity of specification formalisms in the spirit of PDL$_{\mathsf{SL}}$ could be harrowing for the non-initiated user, which could deter researchers from their adoption. Besides, differences between research methodologies in linguistics and computer science, could give way to confusion on the use of the formal language from either domain. For these reasons, the burden of reconciling both lies on PDL$_{\mathsf{SL}}$ itself, which is why special emphasis was given on developing a flexible representation language. With this, resources already in use for linguistic research (like previous annotations) can be represented as PDL$_{\mathsf{SL}}$ formulae, and used by the annotation framework as reference. This is in an effort to provide SL experts with a transparent transition between their linguistic descriptions and the annotation framework, instead of trying to force them into selecting a new contender to the SL transcription arena. However, more work has to be done to achieve this goal.

7.3.1 Perspectives

Future work, from the point of view of multidisciplinary collaboration, is centered in the development of the necessary tools for the automatic conversion of ZeBeDee or HamNoSys descriptions, for example, into PDL$_{\mathsf{SL}}$ formulae. Furthermore, it may also be useful to conduct usability tests over the annotation framework, to detect potential pitfalls induced by the system's design and not necessarily by the formalism itself.

Likewise, the development of interactive tools for the creation of PDL$_{\mathsf{SL}}$ formulae could prove to be a viable alternative to the transformation from other descriptions. In this way, users could define themselves the contents of the primitive sets by way of a visual interface, and an automatic algorithm could use these definitions to calculate the appropriate modeling rules. Similarly, such a system could be fitted with a SL synthesis engine, in order to guide users into the edition of formulae: each formula could be transformed into an utterance representation — following PDL$_{\mathsf{SL}}$ semantic rules — the could, in turn, be used to animate a signing avatar. With this, the expert could see in real-time how the system interprets a given formula, without having to dwell on the details of the formalism.

Finally, more collaboration with SL experts could lead to improvements to the PDL_{SL} syntax and semantics, so as to represent properties beyond the phonetic-phonological level; however, as for now, it might be more important to evaluate which features of the formalism, as-is, are truly useful for linguistic research and which could be simplified to better suit the work in the area.

Bibliography

[Akmajian 2010] Adrian Akmajian, Richard A. Demers, Ann K. Farmer and Robert M. Harnish. *Linguistics: An Introduction to Language and Communication, 6th edition*. The MIT Press, Cambridge, Mass, sixth edition edition edition, March 2010.

[Antia 2001] S. D. Antia and K. H. Kreimeyer. *The role of interpreters in inclusive classrooms*. American Annals of the Deaf, vol. 146, no. 4, pages 355–365, October 2001.

[Antinoro Pizzuto 2010] Elena Antinoro Pizzuto, Isabella Chiari and Paolo Rossini. *Representing sign language: Theoretical, methodological and practical issues*. In Massimo Pettorino, Antonella Giannini, Isabella Chiari and Francesca Dovetto (eds.), Spoken Communication, pages 205–241. Cambridge Scholars Publishing, Newcastle upon Tyne, new edition, August 2010.

[Arikan 2003] Okan Arikan, David A. Forsyth and James F. O'Brien. *Motion Synthesis from Annotations*. In ACM SIGGRAPH 2003 Papers, SIGGRAPH '03, pages 402–408, New York, NY, USA, 2003. ACM.

[Aronoff 2005] Mark Aronoff, Irit Meir and Wendy Sandler. *THE PARADOX OF SIGN LANGUAGE MORPHOLOGY*. vol. 81, no. 2, pages 301–344, 2005.

[ASLFont 2013] ASLFont. *How can you read and write American Sign Language?*, 2013.

[Awad 2009] Charly Awad, Nicolas Courty, Kyle Duarte, Thibaut Le Naour and Sylvie Gibet. *A combined semantic and motion capture database for real-time sign language synthesis*. In Intelligent Virtual Agents, pages 432–438. Springer, 2009.

[Axelsson 2002] Sun Axelsson and Els Van Der Kooij. *Phonological categories in sign language of the netherlands: The role of phonetic implementation and iconicity*. Lot, 2002.

[Bahan 1996] Benjamin Bahan. *Non-Manual Realization of Agreement in American Sign Language*. Doctoral dissertation, Boston University, Boston, MA., 1996.

[Baier 2008] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[Baker 2006] Paul Baker. *Using corpora in discourse analysis*. A&C Black, June 2006.

[Baker 2009a] Anne Baker, Beppie Van den Bogaerde and Bencie Woll. *Methods and procedures in sign language acquisition studies*. In Anne Baker and Bencie Woll (eds.), Sign Language Acquisition, pages 1–49. John Benjamins Publishing, 2009.

[Baker 2009b] Anne Baker and Bencie Woll (eds.). *Sign language acquisition*. John Benjamins Publishing, 2009.

[Bangham 2000] J.A. Bangham, S.J. Cox, R. Elliott, J.R.W. Glauert, I. Marshall, S. Rankov and M. Wells. *Virtual signing: capture, animation, storage and transmission-an overview of the ViSiCAST project*. In IEEE Seminar on Speech and Language Processing for Disabled and Elderly People, pages 6/1–6/7, 2000.

[Banko 2001] Michele Banko and Eric Brill. *Mitigating the Paucity-of-data Problem: Exploring the Effect of Training Corpus Size on Classifier Performance for Natural Language Processing*. In Proceedings of the First International Conference on Human Language Technology Research, HLT '01, pages 1–5, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

[Barbulescu 2004] Laura Barbulescu, Jean-Paul Watson, L. Darrell Whitley and Adele E. Howe. *Scheduling Space–Ground Communications for the Air Force Satellite Control Network*. Journal of Scheduling, vol. 7, no. 1, pages 7–34, January 2004.

[Barendsen 1994] Henk Barendregt Erik Barendsen. *Introduction to lambda calculus*. Nordström, Bengt - Department of Computer Science and Engineering, Chalmers University of Technology, 1994.

[Battison 1974] Robbin Battison. *Phonological deletion in American sign language*. Sign language studies, vol. 5, no. 1, pages 1–19, 1974.

[Battison 1978] Robbin Battison. *Lexical borrowing in american sign language.* Linstok Press, June 1978.

[Bauer 2000] B. Bauer and H. Hienz. *Relevant features for video-based continuous sign language recognition*. In Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000. Proceedings, pages 440–445, 2000.

[Bauer 2002a] B. Bauer and K.-F. Kraiss. *Video-based sign recognition using self-organizing subunits*. In 16th International Conference on Pattern Recognition, 2002. Proceedings, volume 2, pages 434–437 vol.2, 2002.

[Bauer 2002b] Britta Bauer and Kraiss Karl-Friedrich. *Towards an Automatic Sign Language Recognition System Using Subunits*. In Ipke Wachsmuth and Timo Sowa (eds.), Gesture and Sign Language in Human-Computer Interaction, number 2298 de Lecture Notes in Computer Science, pages 64–75. Springer Berlin Heidelberg, 2002.

[Bauland 2009] Michael Bauland, Martin Mundhenk, Thomas Schneider, Henning Schnoor, Ilka Schnoor and Heribert Vollmer. *The Tractability of Model-checking for LTL: The Good, the Bad, and the Ugly Fragments.* Electronic Notes in Theoretical Computer Science, vol. 231, pages 277–292, March 2009.

[Baynton 1995] Douglas Baynton. *Savages and deaf-mutes.* Evolutionary theory and the campaign against sign language. J. Anthropol. Stud. Hum. Mov, vol. 8, no. 4, pages 139–72, 1995.

[Beckner 2009] Clay Beckner, Richard Blythe, Joan Bybee, Morten H. Christiansen, William Croft, Nick C. Ellis, John Holland, Jinyun Ke, Diane Larsen-Freeman and Tom Schoenemann. *Language Is a Complex Adaptive System: Position Paper.* Language Learning, vol. 59, pages 1–26, December 2009.

[Bentley 2001] B. Bentley. *Validating the Intel(R) Pentium(R) 4 microprocessor.* In Design Automation Conference, 2001. Proceedings, pages 244–248, 2001.

[Bergstra 1991] Jan A. Bergstra and Loe M. G. Feijs. *Algebraic methods II: Theory, tools and applications.* Springer Science & Business Media, 1991.

[Biber 1998] Douglas Biber, Susan Conrad and Randi Reppen. *Corpus linguistics: Investigating language structure and use.* Cambridge University Press, April 1998.

[Biemann 2007] Chris Biemann, Gerhard Heyer, Uwe Quasthoff and Matthias Richter. *The Leipzig corpora collection-monolingual corpora of standard size.* Proceedings of Corpus Linguistic 2007, 2007.

[Boer 2012] Frank S. de Boer, Reiner Hähnle, Einar Broch Johnsen, Rudolf Schlatte and Peter Y. H. Wong. *Formal Modeling of Resource Management for Cloud Architectures: An Industrial Case Study.* In Flavio De Paoli, Ernesto Pimentel and Gianluigi Zavattaro (eds.), Service-Oriented and Cloud Computing, number 7592 de Lecture Notes in Computer Science, pages 91–106. Springer Berlin Heidelberg, 2012.

[Booij 2005] G. E. Booij. *The grammar of words an introduction to linguistic morphology /.* Oxford University Press„ 2005.

[Borgia 2015] Fabrizio Borgia. *Informatisation d'une forme graphique des Langues des Signes : application au système d'écriture SignWriting.* PhD thesis, University of Toulouse, Toulouse, France, March 2015.

[Boulares 2012] Mehrez Boulares and Mohamed Jemni. *Mobile Sign Language Translation System for Deaf Community.* In Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, W4A '12, pages 37:1–37:4, New York, NY, USA, 2012. ACM.

[Bowen 1993] J. Bowen and V. Stavridou. *Safety-critical systems, formal methods and standards.* Software Engineering Journal, vol. 8, no. 4, pages 189–209, July 1993.

[Bowker 2002] Lynne Bowker and Jennifer Pearson. *Working with specialized language: A practical guide to using corpora.* Routledge, September 2002.

[Braffort 2004] A. Braffort, A. Choisier, C. Collet, P. Dalle, F. Gianni, F. Lenseigne and J. Segouat. *Toward an annotation software for video of Sign Language, including image processing tools and signing space modelling.* In 4 th International Conference on Language Resources and Evaluation, Lisbonne, pages 201–203, 2004.

[Braffort 2006] Annelies Braffort and Fanch Lejeune. *Spatialised Semantic Relations in French Sign Language: Toward a Computational Modelling.* In Sylvie Gibet, Nicolas Courty and Jean-François Kamp (eds.), Gesture in Human-Computer Interaction and Simulation, number 3881 de Lecture Notes in Computer Science, pages 37–48. Springer Berlin Heidelberg, 2006.

[Braffort 2008] Annelies Braffort and Patrice Dalle. *Sign language applications: preliminary modeling.* Universal Access in the Information Society, vol. 6, no. 4, pages 393–404, 2008.

[Braffort 2010] Annelies Braffort, Laurence Bolot, Emilie Chételat-Pelé, Annick Choisier, Maxime Delorme, Michael Filhol, Jérémie Segouat, Cyril Verrecchia, Flora Badin and Nadège Devos. *Sign Language Corpora for Analysis, Processing and Evaluation.* In LREC, 2010.

[Brashear 2003] Helene Brashear, Thad Starner, Paul Lukowicz and Holger Junker. *Using Multiple Sensors for Mobile Sign Language Recognition.* In Proceedings. Seventh IEEE International Symposium on Wearable Computers., October 2003.

[Bray 1997] Tim Bray, Jean Paoli and C. M. Sperberg-McQueen. *Extensible Markup Language.* World Wide Web J., vol. 2, no. 4, pages 29–66, 1997.

[Brill 2003] Eric Brill. *Processing Natural Language without Natural Language Processing.* In Alexander Gelbukh (ed.), Computational Linguistics and Intelligent Text Processing, number 2588 de Lecture Notes in Computer Science, pages 360–369. Springer Berlin Heidelberg, 2003.

[Brugman 2002] Hennie Brugman, Peter Wittenburg, Stephen C. Levinson and Sotaro Kita. *Multimodal annotations in gesture and sign language studies.* In M. Rodriguez González and C. Paz Suárez Araujo (eds.), Third international conference on language resources and evaluation, pages 176–182. European Language Resources Association., 2002.

[Brugman 2004] Hennie Brugman, Onno Crasborn and Albert Russel. *Collaborative Annotation of Sign Language Data with Peer-to-Peer Technology.* In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04), Lisbon, Portugal, 2004. European Language Resources Association (ELRA).

[Buehler 2008] P. Buehler, M. Everingham, D. P. Huttenlocher and A. Zisserman. *Long term arm and hand tracking for continuous sign language TV broadcasts.* Proceedings of the 19th British Machine Vision Conference, September 2008.

[Buehler 2009] P. Buehler, A. Zisserman and M. Everingham. *Learning sign language by watching TV (using weakly aligned subtitles).* In IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009, pages 2961–2968, June 2009.

[Bungerot 2008] Jan Bungerot, Daniel Stein, Philippe Dreuw, Hermann Ney, Sara Morrissey, Andy Way and Lynette van Zijl. *The ATIS sign language corpus.* In LREC 2008 - Sixth International Conference on Language Resources and Evaluation, Marrakech, Morocco, 2008.

[Börstell 2014] Carl Börstell, Johanna Mesch and Lars Wallin. *Segmenting the Swedish Sign Language corpus : On the possibilities of using visual cues as a basis for syntactic segmentation.* In 9th International Conference on Language Resources and Evaluation, LREC 2014, pages 7–10, Reykjavik, Iceland, 2014. ELRA.

[Calderon 2000] Rosemary Calderon. *Parental Involvement in Deaf Children's Education Programs as a Predictor of Child's Language, Early Reading, and Social-Emotional Development.* Journal of Deaf Studies and Deaf Education, vol. 5, no. 2, pages 140–155, April 2000.

[Cerney 2004] Brian Cerney. *Deaf history notes.* Hand and Mind Publishing, Rochester, NY, USA, November 2004.

[Chai 2013] Xiujuan Chai, Guang Li, Yushun Lin, Zhihao Xu, Yili Tang, Xilin Chen and Ming Zhou. *Sign language recognition and translation with kinect.* IEEE Conf. on AFGR, 2013.

[Chang 1997] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic logic and mechanical theorem proving.* Academic Press, Inc., 1st edition, 1997.

[Chao 2004] Shih-Pin Chao, Chih-Yi Chiu, Shi-Nine Yang and Tsang-Gang Lin. *Tai Chi Synthesizer: A Motion Synthesis Framework Based on Key-postures and Motion Instructions: Research Articles.* Comput. Animat. Virtual Worlds, vol. 15, no. 3-4, pages 259–268, July 2004.

[Chellas 1980] Brian F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge Eng. ; New York, February 1980.

[Chen Pichler 2010] Deborah Chen Pichler, Julie A. Hochgesang, Diane Lillo-Martin and Ronice Müller de Quadros. *Conventions for sign and speech transcription of child bimodal bilingual corpora in ELAN.* LIA, vol. 1, no. 1, pages 11–40, 2010.

[Chippendale 2006] P. Chippendale. *Towards Automatic Body Language Annotation.* In 7th International Conference on Automatic Face and Gesture Recognition, 2006. FGR 2006, pages 487–492, 2006.

[Chiu 2007] Yu-Hsien Chiu, Chung-Hsien Wu, Hung-Yu Su and Chih-Jen Cheng. *Joint Optimization of Word Alignment and Epenthesis Generation for Chinese to Taiwanese Sign Synthesis.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 1, pages 28–39, 2007.

[Chomsky 1957] Noam Chomsky. *Syntactic structures.* Mouton & Co., The Hague, 1957. Reprinted 1985 by Springer, Berlin and New York.

[Clarke 1986] E. M. Clarke, E. A. Emerson and A. P. Sistla. *Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications.* ACM Trans. Program. Lang. Syst., vol. 8, no. 2, pages 244–263, 1986.

[Collet 2010] Christophe Collet, Matilde Gonzalez and Fabien Milachon. *Distributed system architecture for assisted annotation of video corpora.* In International workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies (LREC), pages 49–52, Valletta, Malta, 2010.

[Companys 2004] Monica Companys. *ABC...LSF : Dictionnaire visuel bilingue.* Monica Companys, 2004.

[Contini-Morava 2000] Ellen Contini-Morava and Yishai Tobin. *Between grammar and lexicon.* John Benjamins Publishing, 2000.

[Cooper 2007] Helen Cooper and Richard Bowden. *Large Lexicon Detection of Sign Language.* In Proceedings of the 2007 IEEE International Conference on Human-computer Interaction, HCI'07, pages 88–97, Berlin, Heidelberg, 2007. Springer-Verlag.

[Cooper 2011] Helen Cooper, Brian Holt and Richard Bowden. *Sign Language Recognition.* In Thomas B. Moeslund, Adrian Hilton, Volker Krüger and Leonid Sigal (eds.), Visual Analysis of Humans, pages 539–562. Springer London, 2011.

[Corina 1993] David Corina and Wendy Sandler. *On the Nature of Phonological Structure in Sign Language.* Phonology, vol. 10, no. 2, pages 165–207, January 1993.

[Cox 2002] Stephen Cox, Michael Lincoln, Judy Tryggvason, Melanie Nakisa, Mark Wells, Marcus Tutt and Sanja Abbott. *Tessa, a System to Aid Communication with Deaf People.* In Proceedings of the Fifth International ACM Conference on Assistive Technologies, Assets '02, pages 205–212, New York, NY, USA, 2002. ACM.

[Craigen 1993] Dan H. Craigen, Susan L. Gerhart and Theodore J. Ralston. *An International Survey of Industrial Applications of Formal Methods.* Naval Research Lab, Washington DC. Volume 2. Case Studies. Technical report, September 1993.

[Crasborn 2008a] Onno Crasborn and Han Sloetjes. *Enhanced ELAN functionality for sign language corpora.* In Proceedings of the 3rd Workshop on the Representation and Processing of Sign Languages: Construction and Exploitation of Sign Language Corpora, pages 39–43, Paris, France, 2008.

[Crasborn 2008b] Onno Crasborn, Els Van Der Kooij, Dafydd Waters, Bencie Woll and Johanna Mesch. *Frequency distribution and spreading behavior of different types of mouth actions in three sign languages.* Sign Language & Linguistics, vol. 11, no. 1, pages 45–67, 2008.

[Crasborn 2008c] Onno Crasborn and Inge Zwitserlood. *The Corpus NGT: an online corpus for professionals and laymen.* In Construction and exploitation of sign language corpora. 3rd Workshop on the Representation and Processing of Sign Languages, pages 44–49. ELDA, Paris, France, 2008.

[Cuxac 2000] Christian Cuxac. *La langue des signes française (LSF): les voies de l'iconocité.* Number 15-16. Ophrys, 2000.

[Cuxac 2007a] Christian Cuxac and Patrice Dalle. *Problématique des chercheurs en traitement automatique des langues des signes*, volume 48 of *Traitement Automatique des Langues.* Lavoisier, http://www.editions-hermes.fr/, October 2007.

[Cuxac 2007b] Christian Cuxac and Marie-Anne Sallandre. *Iconicity and arbitrariness in French sign language – highly iconic structures, degenerated iconicity and diagrammatic iconicity.* In Elena Pizzuto, Paola Pietrandrea and Raffaele Simone (eds.), Verbal and Signed Languages: Comparing Structures, Constructs and Methodologies, pages 13–33. Walter de Gruyter, 2007.

[Dalle 2006] Patrice Dalle. *High level models for sign language analysis by a vision system.* In Workshop on the Representation and Processing of Sign Language : Lexicographic Matters and Didactic Scenarios (LREC), pages 17–20, Genoa, Italy, 2006.

[Danecki 1985] Ryszard Danecki. *Nondeterministic Propositional Dynamic Logic with intersection is decidable.* In Andrzej Skowron (ed.), Computation Theory, number 208 de Lecture Notes in Computer Science, pages 34–53. Springer Berlin Heidelberg, 1985.

[Daniels 1996] Marilyn Daniels. *Seeing Language: The Effect of Sign Language on Vocabulary Development in Young Hearing Children.* Child Study Journal, vol. 26, no. 3, page 193, September 1996.

[Davis 1958] Martin Davis. *Computability & Unsolvability*, volume 52. Dover, 1958.

[Ding 2009] Liya Ding and Aleix M. Martinez. *Modelling and recognition of the linguistic components in American Sign Language.* Image and Vision Computing, vol. 27, no. 12, pages 1826–1844, November 2009.

[Dorner 1994] Brigitte Dorner and Eli Hagen. *Towards an American Sign Language interface.* Artificial Intelligence Review, vol. 8, no. 2-3, pages 235–253, March 1994.

[Dreuw 2008a] Philippe Dreuw and Hermann Ney. *Towards Automatic Sign Language Annotation for the ELAN Tool.* In LREC Workshop on the Representation and Processing of Sign Languages: Construction and Exploitation of Sign Language Corpora, pages 50–53, Marrakech, Morocco, June 2008.

[Dreuw 2008b] Philippe Dreuw, Daniel Stein, Thomas Deselaers, David Rybach, Morteza Zahedi, Jan Bungeroth and Hermann Ney. *Spoken language processing techniques for sign language recognition and translation.* Technology and Disability, vol. 20, pages 121–133, 2008.

[Duarte 2010] Kyle Duarte and Sylvie Gibet. *Heterogeneous Data Sources for Signed Language Analysis and Synthesis: The SignCom Project.* In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner and Daniel Tapias (eds.), Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, 2010. European Language Resources Association (ELRA).

[Dubot 2012] Rémi Dubot and Christophe Collet. *Improvements of the Distributed Architecture for Assisted Annotation of video corpora (poster).* In Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon, pages 27–30, Istambul, Turkey, 2012. European Language Resources Association (ELRA).

[Duchenne 2009] O. Duchenne, I. Laptev, J. Sivic, F. Bach and J. Ponce. *Automatic annotation of human actions in video.* In 2009 IEEE 12th International Conference on Computer Vision, pages 1491–1498, Kyoto, Japan, 2009.

[Efthimiou 2012a] Eleni Efthimiou, Stavroula-Evita Fotinea, T. Hanke, J. Glauert, R. Bowden, A. Braffort, P. Maragos and F. Lefebvre-Albaret. *Sign Language technologies and resources of the Dicta-Sign project.* In Proc. of the 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon. Satellite Workshop to the eighth International Conference on Language Resources and Evaluation (LREC-2012), pages 37–44, Istanbul, Turkey, May 2012.

[Efthimiou 2012b] Eleni Efthimiou, Stavroula-Evita Fotinea, Thomas Hanke, John Glauert, Richard Bowden, Annelies Braffort, Christophe Collet, Petros Maragos and François Lefebvre-Albaret. *The Dicta-Sign Wiki: Enabling Web Communication for the Deaf.* In Klaus Miesenberger, Arthur Karshmer, Petr Penaz and Wolfgang Zagler (eds.), Computers Helping People with Special Needs, number

7383 de Lecture Notes in Computer Science, pages 205–212. Springer Berlin Heidelberg, 2012.

[Elliott 2004] R. Elliott, J. R. W. Glauert, V. Jennings and J. R. Kennaway. *An Overview of the SiGML Notation and SiGMLSigning Software System.* In Sign Language Processing Satellite Workshop of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, pages 98–104, Lisbon, May 2004.

[Elliott 2007] R. Elliott, J. R. W. Glauert, J. R. Kennaway, I. Marshall and E. Safar. *Linguistic modelling and language-processing technologies for Avatar-based sign language presentation.* Universal Access in the Information Society, vol. 6, no. 4, pages 375–391, October 2007.

[Elliott 2012] R. Elliott, H. M. Cooper, E. J. Ong, J. Glauert, R. Bowden and F. Lefebvre-Albaret. *Search-By-Example in Multilingual Sign Language Databases.* In 2nd International Workshop on Sign Language Translation and Avatar Technology (SLTAT), Dundee, UK, June 2012.

[Emerson 1997] E. Allen Emerson. *Model checking and the Mu-calculus.* In DIMACS Series in Discrete Mathematics, pages 185–214. American Mathematical Society, 1997.

[Emmorey 2001] Karen Emmorey. *Language, cognition, and the brain: Insights from sign language research.* Lawrence Erlbaum Associates Publishers, Mahwah, NJ, US, November 2001.

[Emmorey 2003] Karen Emmorey. *Space on hand: The exploitation of signing space to illustrate abstract thought.* In Merideth Gattis (ed.), Spatial Schemas and Abstract Thought, pages 147–174. MIT Press, Cambridge, MA, January 2003.

[Emmorey 2009] Karen Emmorey and Stephen McCullough. *The bimodal bilingual brain: Effects of sign language experience.* Brain and language, vol. 109, no. 2-3, pages 124–132, 2009.

[Enerstvedt 1996] Regi Enerstvedt. *Legacy of the past: some aspects in the history of blind education, deaf education, deaf-blind education with emphasis on the time before 1900.* Forlaget Nord-Press, Dronninglund, Danmark, 1996.

[Engberg-Pedersen 1993] Elisabeth Engberg-Pedersen. *Space in danish sign language: The semantics and morphosyntax of the use of space in a visual language.* Gallaudet University Press, 1993.

[Engberg-Pedersen 2003] Elisabeth Engberg-Pedersen. *From pointing to reference and predication: pointing signs, eyegaze, and head and body orientation in Danish Sign Language.* In Sotaro Kita (ed.), Pointing: Where Language, Culture, and Cognition Meet, pages 269–293. Psychology Press, June 2003.

[Erdem 2002] U.M. Erdem and S. Sclaroff. *Automatic detection of relevant head gestures in American Sign Language communication.* In 16th International Conference on Pattern Recognition, 2002. Proceedings, volume 1, pages 460–463 vol.1, Quebec, Canada, 2002.

[Erenshteyn 1996] Roman Erenshteyn and Pavel Laskov. *A Multi-Stage Approach To Fingerspelling And Gesture Recognition.* In Proceedings of the Workshop on the Integration of Gesture in Language and Speech, pages 185–194. Academic Press, Inc, 1996.

[Eriks-Brophy 2004] Alice Eriks-Brophy. *Outcomes of Auditory-Verbal Therapy: A Review of the Evidence and a Call for Action.* Volta Review, vol. 104, no. 1, pages 21–35, January 2004.

[Eriksson 1998] Per Eriksson. *The History of Deaf People: A Source Book.* Daufr, Örebro, 1998.

[Fang 2001] Gaolin Fang, Wen Gao and Jiyong Ma. *Signer-independent sign language recognition based on SOFM/HMM.* In IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings, pages 90–95, Washington, D.C., USA, 2001.

[Fang 2007] Gaolin Fang, Wen Gao and Debin Zhao. *Large-Vocabulary Continuous Sign Language Recognition Based on Transition-Movement Models.* IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 37, no. 1, pages 1–9, 2007.

[Fanghella 2012] Julia Fanghella, Leah Geer, Jonathan Henner, Julie Hochgesang, Diane Lillo-Martin, Gaurav Mathur, Gene Mirus and Pedro Pascual-Villaneuva. *Linking an ID-Gloss database of ASL with child language corpora.* In Proceedings of the 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon., pages 57–62, Istanbul, Turkey, May 2012.

[Filhol 2007] Michael Filhol, Annelies Braffort and Laurence Bolot. *Signing avatar: Say hello to Elsi!* In Gesture-Based Human-Computer Interaction and Simulation: 7th International Gesture Workshop (GW 2007), pages 44–45, Lisbon, Portugal, 2007.

[Filhol 2009] Michael Filhol. *Zebedee: a lexical description model for Sign Language synthesis.* LIMSI-CNRS, Orsay, 2009.

[Filhol 2012a] Michael Filhol. *Combining Two Synchronisation Methods in a Linguistic Model to Describe Sign Language.* In Eleni Efthimiou, Georgios Kouroupetroglou and Stavroula-Evita Fotinea (eds.), Gesture and Sign Language in Human-Computer Interaction and Embodied Communication, number 7206 de Lecture Notes in Computer Science, pages 194–203. Springer Berlin Heidelberg, 2012.

[Filhol 2012b] Michael Filhol and Annelies Braffort. *Méthodologie d'exploration de corpus et de formalisation de règles grammaticales pour les langues des signes.* In Actes de la 19e conférence sur le Traitement Automatique des Langues Naturelles, pages 375–382, Grenoble, France, June 2012. Association pour le Traitement Automatique des Langues.

[Fischer 1979] Michael J. Fischer and Richard E. Ladner. *Propositional dynamic logic of regular programs.* Journal of Computer and System Sciences, vol. 18, no. 2, pages 194–211, 1979.

[Flood 2002] Cecilia Mary Flood. *How Do Deaf and Hard of Hearing Students Experience Learning to Write Using Signwriting, a Way to Read and Write Signs?* The University of New Mexico, 2002.

[Forster 2010] Jens Forster, Daniel Stein, Ellen Ormel, Onno Crasborn and Hermann Ney. *Best Practice for Sign Language Data Collections Regarding the Needs of Data-Driven Recognition and Translation.* In 4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies, pages 92–97, Valletta, Malta, May 2010.

[Forster 2012] Jens Forster, Christoph Schmidt, Thomas Hoyoux, Oscar Koller, Uwe Zelle, Justus Piater and Hermann Ney. *RWTH-PHOENIX-Weather: A Large Vocabulary Sign Language Recognition and Translation Corpus.* In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk and Stelios Piperidis (eds.), Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), pages 3785–3789, Istanbul, Turkey, 2012. European Language Resources Association (ELRA).

[Forster 2014] Jens Forster, Christoph Schmidt, Oscar Koller, Martin Bellgardt and Hermann Ney. *Extensions of the Sign Language Recognition and Translation Corpus RWTH-PHOENIX-Weather.* In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), pages 1911–1916. European Language Resources Association (ELRA), Reykjavik, Iceland, 2014.

[Fotinea 2012] Stavroula-Evita Fotinea, Eleni Efthimiou and Athanasia-Lida Dimou. *Sign Language Computer-Aided Education: Exploiting GSL Resources and Technologies for Web Deaf Communication.* In Klaus Miesenberger, Arthur Karshmer, Petr Penaz and Wolfgang Zagler (eds.), Computers Helping People with Special Needs, number 7383 de Lecture Notes in Computer Science, pages 237–244. Springer Berlin Heidelberg, 2012.

[Frawley 2006] William Frawley, Erin Eschenroeder, Sarah Mills and Thao Nguyen. *The Expression of Modality.* Walter de Gruyter, 2006.

[Frishberg 1975] Nancy Frishberg. *Arbitrariness and Iconicity: Historical Change in American Sign Language*. Language, vol. 51, no. 3, pages 696–719, September 1975.

[Gianni 2009] F. Gianni, C. Collet and P. Dalle. *Robust tracking for processing of videos of communication's gestures*. Gesture-Based Human-Computer Interaction and Simulation, pages 93–101, 2009.

[Gibbon 2009] Dafydd Gibbon. *Gesture Theory is Linguistics: On Modelling Multimodality as Prosody*. In Olivia Kwong (ed.), PACLIC, pages 9–18. City University of Hong Kong Press, 2009.

[Goldin-Meadow 1998] Susan Goldin-Meadow and Carolyn Mylander. *Spontaneous sign systems created by deaf children in two cultures*. Nature, vol. 391, no. 6664, pages 279–281, January 1998.

[Gonzalez 2011] Matilde Gonzalez and Christophe Collet. *Robust body parts tracking using particle filter and dynamic template*. In 2011 18th IEEE International Conference on Image Processing (ICIP), pages 529 –532, Brussels, Belgium, September 2011.

[Gonzalez 2012] Matilde Gonzalez and Christophe Collet. *Sign Segmentation Using Dynamics and Hand Configuration for Semi-automatic Annotation of Sign Language Corpora*. In Gesture and Sign Language in Human-Computer Interaction and Embodied Communication, LNCS, pages 204–215. Springer, 2012.

[Goodwyn 2000] Susan W. Goodwyn, Linda P. Acredolo and Catherine A. Brown. *Impact of Symbolic Gesturing on Early Language Development*. Journal of Nonverbal Behavior, vol. 24, no. 2, pages 81–103, June 2000.

[Green 2004] R.D. Green and Ling Guan. *Quantifying and recognizing human movement patterns from monocular video Images-part I: a new framework for modeling human motion*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 2, pages 179–190, February 2004.

[Grobel 1997] K. Grobel and M. Assan. *Isolated sign language recognition using hidden Markov models*. In , 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation, volume 1, pages 162–167 vol.1, October 1997.

[Guttag 1978] J. V. Guttag and J. J. Horning. *The algebraic specification of abstract data types*. Acta Informatica, vol. 10, no. 1, pages 27–52, March 1978.

[Guttag 1993] John V. Guttag and James J. Horning. *Larch: Languages and Tools for Formal Specification*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[Hagan 2002] Martin T. Hagan, Howard B. Demuth and Mark H. Beale. *Neural Network Design*. Campus Pub. Service, University of Colorado Bookstore, January 2002.

[Halevy 2009] Alon Halevy, Peter Norvig and Fernando Pereira. *The Unreasonable Effectiveness of Data.* IEEE Intelligent Systems, vol. 24, no. 2, pages 8–12, 2009.

[Hall 1996] A. Hall. *Using formal methods to develop an ATC information system.* IEEE Software, vol. 13, no. 2, pages 66–76, March 1996.

[Hamada 2004] Y. Hamada, N. Shimada and Y. Shirai. *Hand shape estimation under complex backgrounds for sign language recognition.* In Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings, pages 589–594, Seoul, Korea, May 2004.

[Han 2009] Junwei Han, George Awad and Alistair Sutherland. *Modelling and segmenting subunits for sign language recognition based on hand motion analysis.* Pattern Recognition Letters, vol. 30, no. 6, pages 623–633, April 2009.

[Hanke 2001] Thomas Hanke. *Sign language transcription with syncWRITER.* Sign Language & Linguistics, vol. 4, no. 1, pages 275–283, 2001.

[Hanke 2002] Thomas Hanke. *iLex-A tool for Sign Language Lexicography and Corpus Analysis.* In LREC, pages 923–926, Istanbul, Turkey, 2002. European Language Resources Association.

[Hanke 2004] Thomas Hanke. *HamNoSys-representing sign language data in language resources and language processing contexts.* In 5th workshop on the representation and processing of sign languages : interactions between corpus and lexicon : workshop proceedings (LREC), volume 4, pages 1–6, Lisbon, Portugal, 2004. European Language Resources Association.

[Hanke 2010] Thomas Hanke, Lutz König, Sven Wagner and Silke Matthes. *DGS Corpus & Dicta-Sign: The Hamburg Studio Setup.* In 4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies (CSLT 2010), Valletta, Malta, pages 106–110, 2010.

[Hartel 2005] Frank W. Hartel, Sherri de Coronado, Robert Dionne, Gilberto Fragoso and Jennifer Golbeck. *Modeling a description logic vocabulary for cancer research.* Journal of Biomedical Informatics, vol. 38, no. 2, pages 114–129, April 2005.

[Hayes 2008] Bruce Hayes. *Introductory phonology.* Wiley-Blackwell, 1 edition edition, 2008.

[Heloir 2015] Alexis Heloir and Fabrizio Nunnari. *Toward an intuitive sign language animation authoring system for the deaf.* Universal Access in the Information Society, pages 1–11, May 2015.

[Hintikka 1962] Jaakko Hintikka. *Knowledge and belief.* Ithaca, N.Y.,Cornell University Press, 1962.

[Hoare 1985] C.A. R. Hoare. *Communicating sequential processes.* Prentice-Hall, Englewood Cliffs, N.J, 1985.

[Hoek 2003] Wiebe van der Hoek and Michael Wooldridge. *Cooperation, Knowledge, and Time: Alternating-time Temporal Epistemic Logic and its Applications*. Studia Logica, vol. 75, no. 1, pages 125–157, October 2003.

[Hogan 2008] Sarah Hogan, Jacqueline Stokes, Catherine White, Elizabeth Tyszkiewicz and Alexandra Woolgar. *An evaluation of Auditory Verbal therapy using the rate of early language development as an outcome measure*. Deafness & Education International, vol. 10, no. 3, pages 143–167, September 2008.

[Hoiting 2003] Nini Hoiting and Dan I. Slobin. *Transcription as a tool for understanding: The Berkeley Transcription System for sign language research (BTS*. In In G. Morgan & B. Woll, (Eds.), Directions in sign language acquisition, pages 55–75. Amsterdam/Philadelphia: John Benjamins, 2003.

[Holloway 1997] C.M. Holloway. *Why engineers should consider formal methods*. In , AIAA/IEEE Digital Avionics Systems Conference, 1997. 16th DASC, volume 1, pages 1.3–16–22 vol.1, October 1997.

[Hong 2006] S. Hong. *Agreement verbs in Korean Sign Language (KSL)*. In Ninth conference on Theoretical Issues in Sign Language Research (TISLR 9), Floriano polis, Brazil, pages 168–188, Florianopolis, Brazil, 2006.

[Huang 2004] Zhisheng Huang, Anton Eliëns and Cees Visser. *STEP: a Scripting Language for Embodied Agents*. In Helmut Prendinger and Mitsuru Ishizuka (eds.), Life-Like Characters, Cognitive Technologies, pages 87–109. Springer Berlin Heidelberg, 2004.

[Huenerfauth 2006] Matt Huenerfauth. *Generating American Sign Language classifier predicates for English-to-ASL machine translation*. pages 1–296, 2006.

[Huenerfauth 2007] Matt Huenerfauth, Liming Zhao, Erdan Gu and Jan Allbeck. *Evaluating American Sign Language Generation Through the Participation of Native ASL Signers*. In Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility, Assets '07, pages 211–218, New York, NY, USA, 2007. ACM.

[Isard 1998] Michael Isard and Andrew Blake. *CONDENSATION—Conditional Density Propagation for Visual Tracking*. International Journal of Computer Vision, vol. 29, no. 1, pages 5–28, August 1998.

[Jacky 1996] Jonathan Jacky. *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, New York, NY, USA, 1996.

[Jenks 2011] Christopher Joseph Jenks. *Transcribing talk and interaction: Issues in the representation of communication data*. John Benjamins Publishing, 2011.

[Johnson 1989] Robert E. Johnson and And Others. *Unlocking the Curriculum: Principles for Achieving Access in Deaf Education.* Research at Gallaudet, Galaudet Research Institute, January 1989.

[Johnson 2011] Robert E. Johnson and Scott K. Liddell. *A Segmental Framework for Representing Signs Phonetically.* vol. 11, no. 3, pages 408–463, 2011.

[Johnston 2006] Trevor Johnston and Adam Schembri. *Issues in the creation of a digital archive of a signed language.* In Linda Barwick and Nicholas Thieberger (eds.), Sustainable Data from Digital Fieldwork: Proceedings of the Conference Held at the University of Sydney, 4-6 December 2006, pages 7–16. Sydney University Press, January 2006.

[Johnston 2010] Trevor Johnston. *From archive to corpus: transcription and annotation in the creation of signed language corpora.* International journal of corpus linguistics, vol. 15, no. 1, pages 106–131, 2010.

[Johnston 2011] Trevor Johnston and L. De Beuzeville. *Auslan Corpus annotation guidelines.* Manuscript, Centre for Language Sciences, Department of Linguistics, Macquarie University, Sydney, Australia. Available online: http://www. auslan. org. au/video/upload/attachments/AuslanCorpusAnnotationGuid elines30November2011. pdf, 2011.

[Jones 1980] Cliff B. Jones. *Software Development: A Rigorous Approach.* Prentice Hall, Englewood Cliffs, N.J, first edition edition edition, February 1980.

[Jones 1990] Cliff B. Jones. *Systematic Software Development Using VDM (2nd Ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[Jung 1994] Moon-Ryul Jung, Norman Badler and Tsukasa Noma. *Animated human agents with motion planning capability for 3D-space postural goals.* The Journal of Visualization and Computer Animation, vol. 5, no. 4, pages 225–246, October 1994.

[Kadous 1996] Mohammed Waleed Kadous and Computer Science Engineering. *Machine Recognition of Auslan Signs Using PowerGloves: Towards Large-Lexicon Recognition of Sign Language.* In Proceedings of the Workshop on the Integration of Gesture in Language and Speech, pages 165–174, Wilmington, DE, USA, 1996.

[Kamp 1993] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Springer Science & Business Media, July 1993.

[Karpouzis 2007] K. Karpouzis, G. Caridakis, S.-E. Fotinea and E. Efthimiou. *Educational resources and implementation of a Greek sign language synthesis architecture.* Computers & Education, vol. 49, no. 1, pages 54 – 74, 2007. Web3D Technologies in Learning, Education and Training.

[Kegl 1976] Judy Anne Kegl and R. BRING-WILBUR. *When Does Structure Stop and Style Begin? Syntax, Morphology, and Phonology v.-s. Stylistic Variation in American Sign Language.* In Papers from the... Regional Meeting. Chicago Ling. Soc. Chicago, Ill., pages 376–396, 1976.

[Keller 1976] Robert M. Keller. *Formal Verification of Parallel Programs.* Commun. ACM, vol. 19, no. 7, pages 371–384, July 1976.

[Kelly 2009] Daniel Kelly, J. McDonald and C. Markham. *Recognizing Spatiotemporal Gestures and Movement Epenthesis in Sign Language.* In Machine Vision and Image Processing Conference, 2009. IMVIP '09. 13th International Machine Vision and Image Processing Conference, pages 145–150, 2009.

[Kennaway 2007] J. R. Kennaway, J. R. W. Glauert and I. Zwitserlood. *Providing Signed Content on the Internet by Synthesized Animation.* ACM Trans. Comput.-Hum. Interact., vol. 14, no. 3, September 2007.

[Kipp 2011] Michael Kipp, Alexis Heloir and Quan Nguyen. *Sign Language Avatars: Animation and Comprehensibility.* In Hannes Högni Vilhjálmsson, Stefan Kopp, Stacy Marsella and Kristinn R. Thórisson (eds.), Intelligent Virtual Agents, number 6895 de Lecture Notes in Computer Science, pages 113–126. Springer Berlin Heidelberg, January 2011.

[Knight 1997] John C. Knight, Colleen L. DeJong, Matthew S. Gibble and Luis G. Nakano. *Why are Formal Methods Not Used More Widely?* In Fourth NASA Formal Methods Workshop. NASA, Charlottesville, VA, United States, September 1997.

[Koller 2013] O. Koller, H. Ney and R. Bowden. *May the force be with you: Force-aligned signwriting for automatic subunit annotation of corpora.* In 2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), pages 1–6, April 2013.

[Kwiatkowska 2007] M. Kwiatkowska, G. Norman and D. Parker. *Stochastic Model Checking.* In M. Bernardo and J. Hillston (eds.), Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07), volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

[Ladefoged 1996] Peter Ladefoged and Ian Maddieson. *The Sounds of the World's Languages.* Wiley-Blackwell, Oxford, OX, UK ; Cambridge, Mass., USA, 1 edition edition, February 1996.

[Lamport 1989] Leslie Lamport. *A Simple Approach to Specifying Concurrent Systems.* Commun. ACM, vol. 32, no. 1, pages 32–45, January 1989.

[Lange 2006] Martin Lange. *Model checking propositional dynamic logic with all extras.* Journal of Applied Logic, vol. 4, no. 1, pages 39–49, March 2006.

[Lapadat 1999] Judith C. Lapadat and Anne C. Lindsay. *Transcription in Research and Practice: From Standardization of Technique to Interpretive Positionings.* Qualitative Inquiry, vol. 5, no. 1, pages 64–86, March 1999.

[Lebourque 1999] T. Lebourque and S. Gibet. *High level specification and control of communication gestures: the GESSYCA system.* In Computer Animation, 1999. Proceedings, pages 24–35, 1999.

[Leech 1992] Geoffrey Leech. *100 million words of English: the British National Corpus (BNC).* Language Research, vol. 28, no. 1, pages 1–13, 1992.

[Leeson 2006] Lorraine Leeson. *Moving Heads and Moving Hands: Developing a Digital Corpus of Irish Sign Language.* In Information Technology and Telecommunications Conference 2006, Carlow, Ireland, 2006.

[Lefebvre-Albaret 2010] François Lefebvre-Albaret and Patrice Dalle. *Body Posture Estimation in Sign Language Videos.* In Stefan Kopp and Ipke Wachsmuth (eds.), Gesture in Embodied Communication and Human-Computer Interaction, number 5934 de Lecture Notes in Computer Science, pages 289–300. Springer Berlin Heidelberg, 2010.

[Lefebvre-Albaret 2013] François Lefebvre-Albaret, Sylvie Gibet, Ahmed Turki, Ludovic Hamon and Rémi Brun. *Overview of the Sign3D Project High-fidelity 3D recording, indexing and editing of French Sign Language content.* In Third International Symposium on Sign Language Translation and Avatar Technology (SLTAT) 2013, Chicago, United States, October 2013.

[Lejeune 2002] Fanch Lejeune, Annelies Braffort and Desclés Jean-Pierre. *Study on Semantic Representations of French Sign Language Sentences.* In Ipke Wachsmuth and Timo Sowa (eds.), Gesture and Sign Language in Human-Computer Interaction, number 2298 de Lecture Notes in Computer Science, pages 197–201. Springer Berlin Heidelberg, 2002.

[L'Epée 1776] Charles-Michel de L'Epée. *Institution des sourds et muets, par la voie des signes méthodiques; ouvrage qui contient le projet d'une langue universelle, par l'entremise des signes naturels assujettis à une méthode.* Nyon l'ainé, Paris, France, 1776.

[Liang 1998] Rung-Huei Liang and Ming Ouhyoung. *A real-time continuous gesture recognition system for sign language.* In Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998. Proceedings, pages 558–567, April 1998.

[Liddell 1989] S. K. Liddell and R. E. Johnson. *American sign language: The phonological base.* Gallaudet University Press, Washington. DC, 1989.

[Liddell 2003] Scott K. Liddell. *Grammar, Gesture, and Meaning in American Sign Language.* Cambridge University Press, 40 West 20th Street, New York, NY 10011-4211 (Hardbound: ISBN-0-521-81620-3, $70; Paperbound: ISBN-0-521-01650-9, $25). Tel: 212-924-3900 ext 310; Web site: http://www.cambridge.org., March 2003.

[Lieber 2010] Rochelle Lieber. *Introducing morphology.* Cambridge University Press, 1 edition edition, 2010.

[Lillo-Martin 1995] Diane Lillo-Martin. *The point of view predicate in American Sign Language.* In Karen Emmorey and Judy S. Reilly (eds.), Language, Gesture, and Space, pages 155–170. Psychology Press, June 1995.

[Liu 2014] Jingjing Liu, Bo Liu, Shaoting Zhang, Fei Yang, Peng Yang, Dimitris N. Metaxas and Carol Neidle. *Non-manual grammatical marker recognition based on multi-scale, spatio-temporal analysis of head pose and facial expressions.* Image and Vision Computing, vol. 32, no. 10, pages 671–681, October 2014.

[Losson 1998] O. Losson and J.-m Vannobel. *Sign language formal description and synthesis.* INT.JOURNAL OF VIRTUAL REALITY, vol. 3, pages 27–34, 1998.

[Lu 2010] Pengfei Lu and Matt Huenerfauth. *Collecting a motion-capture corpus of American Sign Language for data-driven generation research.* In Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies, pages 89–97. Association for Computational Linguistics, 2010.

[Lyons 1977] John Lyons. *Deixis, space and time.* In Semantics, volume 2. Cambridge University Press, 1977.

[Ma 2000] Jiyong Ma, Wen Gao and Rui Wang. *A Parallel Multistream Model for Integration of Sign Language Recognition and Lip Motion.* In Tieniu Tan, Yuanchun Shi and Wen Gao (eds.), Advances in Multimodal Interfaces — ICMI 2000, number 1948 de Lecture Notes in Computer Science, pages 582–589. Springer Berlin Heidelberg, 2000.

[MacLaughlin 2000] Dawn MacLaughlin, Carol Neidle, Benjamin Bahan and Robert G. Lee. *Morphological Inflections and Syntactic Representations of Person and Number in ASL.* Recherches linguistiques de Vincennes, no. 29, pages 73–100, May 2000.

[Mann 2010] Wolfgang Mann, Chloe R. Marshall, Kathryn Mason and Gary Morgan. *The Acquisition of Sign Language: The Impact of Phonetic Complexity on Phonology.* Language Learning and Development, vol. 6, no. 1, pages 60–86, January 2010.

[Marschark 2001] Marc Marschark, Harry G. Lang and John A. Albertini. *Educating deaf students : From research to practice: From research to practice*. Oxford University Press, November 2001.

[Marshall 2004] Ian Marshall and Eva Sáfár. *Sign language generation in an ALE HPSG*. In Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar, Leuven, pages 189–201, 2004.

[Marshall 2005] Ian Marshall and Eva Safar. *Grammar development for sign language avatar-based synthesis*. In Proceedings HCII, pages 1–10, 2005.

[Martell 2002] Craig H. Martell. *An Extensible, Kinematically-Based Gesture Annotation Scheme*. In Advances in Natural Multimodal Dialogue Systems. Springer Berlin Heidelberg, 2002. Chapter 1 in the book: Advances in Natural Multimodal Dialogue Systems.

[Matthes 2012] Silke Matthes, Thomas Hanke, Anja Regen, Jakob Storz, Satu Worseck, Eleni Efthimiou, Athanasia-Lida Dimou, Annelies Braffort, John Glauert and Eva Safar. *Dicta-Sign–building a multilingual sign language corpus*. In Proc. of the 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon. Satellite Workshop to LREC, pages 23–27, 2012.

[Mayer 1996] Connie Mayer and Gordon Wells. *Can the Linguistic Interdependence Theory Support A Bilingual-Bicultural Model of Literacy Education for Deaf Students?* Journal of Deaf Studies and Deaf Education, vol. 1, no. 2, pages 93–107, March 1996.

[McKee 2000] David McKee and Graeme Kennedy. *Lexical comparison of signs from American, Australian, British and New Zealand sign languages*. The signs of language revisited: An anthology to honor Ursula Bellugi and Edward Klima, pages 49–76, 2000.

[McKee 2002] Rachel Locker McKee and Jemina Napier. *Interpreting into International Sign Pidgin: An analysis*. Journal of Sign Language Linguistics, vol. 5, no. 1, page 333–352, 2002.

[Mehdi 2002] S.A. Mehdi and Y.N. Khan. *Sign language recognition using sensor gloves*. In International Conference on Neural Information Processing, volume 5, pages 2204–2206, Singapore, 2002.

[Meier 2002] Richard P. Meier, Kearsy Cormier and David Quinto-Pozos. *Modality and Structure in Signed and Spoken Languages*. Cambridge University Press, October 2002.

[Meir 1998] Irit Meir. *Syntactic-semantic interaction in Israeli Sign Language verbs: The case of backwards verbs*. Sign Language & Linguistics, vol. 1, no. 1, pages 3–37, 1998.

[Miller 2001] Christopher Miller. *Some reflections on the need for a common sign notation.* Sign Language & Linguistics, vol. 4, no. 1-1, pages 11–28, January 2001.

[Milner 1982] R. Milner. *A calculus of communicating systems.* Springer-Verlag New York, Inc., 1982.

[Moehrmann 2012] Julia Moehrmann and Gunther Heidemann. *Efficient Annotation of Image Data Sets for Computer Vision Applications.* In Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications, VIGTA '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM.

[Moehrmann 2013] Julia Moehrmann and Gunther Heidemann. *Semi-automatic Image Annotation.* In Richard Wilson, Edwin Hancock, Adrian Bors and William Smith (eds.), Computer Analysis of Images and Patterns, number 8048 de Lecture Notes in Computer Science, pages 266–273. Springer Berlin Heidelberg, 2013.

[Monin 2003] Jean-Francois Monin and M. G. Hinchey. *Understanding Formal Methods.* Springer Science & Business Media, 2003.

[Morgan 2003] G. Morgan. *Transcription of child sign language.* Deafness & Education International, vol. 5, no. 3, pages 157–166, 2003.

[Morrissey 2005] Sara Morrissey and Andy Way. *An example-based approach to translating sign language.* In Morrissey, Sara and Way, Andy (2005) An example-based approach to translating sign language. In: Second Workshop on Example-based Machine Translation, 16 September 2005, Phuket, Thailand., Phuket, Thailand, 2005.

[Morrissey 2006] Sara Morrissey and Andy Way. *Lost in translation: the problems of using mainstream MT evaluation metrics for sign language translation.* In Morrissey, Sara and Way, Andy (2006) Lost in translation: the problems of using mainstream MT evaluation metrics for sign language translation. In: SALTMIL 2006 - 5th SALTMIL Workshop on Minority Languages, 23 May 2006, Genoa, Italy., Genoa, Italy, 2006.

[Morrissey 2010] Sara Morrissey, Harold Somers, Robert Smith, Shane Gilchrist and Sandipan Dandapat. *Building a sign language corpus for use in machine translation.* In Morrissey, Sara and Somers, Harold and Smith, Robert and Gilchrist, Shane and Dandapat, Sandipan (2010) Building a sign language corpus for use in machine translation. In: the 4th Workshop on Representation and Processing of Sign Languages: Corpora for Sign Language Technologies, 17 - 23 May 2010, Valetta, Malta., Valetta, Malta, May 2010.

[Moss 2007] Lawrence Moss and Hans Joerg Tiede. *Applications of Modal Logic in Linguistics.* Studies in Logic and Practical Reasoning, pages 1031–1076, 2007.

[Munib 2007] Qutaishat Munib, Moussa Habeeb, Bayan Takruri and Hiba Abed Al-Malik. *American sign language (ASL) recognition based on Hough transform and neural networks.* Expert Systems with Applications, vol. 32, no. 1, pages 24–37, January 2007.

[Müller de Quadros 2014] Ronice Müller de Quadros, Diane Lillo-Martin and Deborah Chen Pichler. *Methodological considerations for the development and use of sign language acquisition corpora.* In Tommaso Raso and Heliana Mello (eds.), Spoken Corpora and Linguistic Studies, volume vii, pages 84–102. John Benjamins Publishing Company, November 2014.

[Nam 1996] Yanghee Nam, Taejeon Korea and KwangYun Wohn. *Recognition of Space-Time Hand-Gestures using Hidden Markov Model.* In ACM Symposium on Virtual Reality Software and Technology, pages 51–58, 1996.

[Nayak 2009] S. Nayak, S. Sarkar and B. Loeding. *Automated extraction of signs from continuous sign language sentences using Iterated Conditional Modes.* In IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009, pages 2583–2590, 2009.

[Neidle 2000] Carol Jan Neidle. *The Syntax of American Sign Language: Functional Categories and Hierarchical Structure.* MIT Press, 2000.

[Neidle 2001] Carol Neidle, Stan Sclaroff and Vassilis Athitsos. *SignStream: A tool for linguistic and computer vision research on visual-gestural language data.* Behavior Research Methods, Instruments, & Computers, vol. 33, no. 3, pages 311–320, August 2001.

[Neidle 2009] C. Neidle, J. Nash, N. Michael and D. Metaxas. *A method for recognition of grammatically significant head movements and facial expressions, developed through use of a linguistically annotated video corpus.* In Proceedings of the Language and Logic Workshop, Formal Approaches to Sign Languages, European Summer School in Logic, Language, and Information (ESSLLI 2009), Bordeaux, France, 2009.

[Nelson 2009] Delroy Nelson. *Using a Signing Avatar as a Sign Language Research Tool.* In Anna Esposito, Amir Hussain, Maria Marinaro and Raffaele Martone (eds.), Multimodal Signals: Cognitive and Algorithmic Issues, number 5398 de Lecture Notes in Computer Science, pages 83–93. Springer Berlin Heidelberg, 2009.

[Neville 1997] Helen J. Neville, Sharon A. Coffey, Donald S. Lawson, Andrew Fischer, Karen Emmorey and Ursula Bellugi. *Neural Systems Mediating American Sign Language: Effects of Sensory Experience and Age of Acquisition.* Brain and Language, vol. 57, no. 3, pages 285–308, May 1997.

[Nickel 2003] Kai Nickel and Rainer Stiefelhagen. *Pointing Gesture Recognition Based on 3D-tracking of Face, Hands and Head Orientation.* In Proceedings of the 5th

International Conference on Multimodal Interfaces, ICMI '03, pages 140–146, New York, NY, USA, 2003. ACM.

[Niewiadomski 2009] Radoslaw Niewiadomski, Elisabetta Bevacqua, Maurizio Mancini and Catherine Pelachaud. *Greta: An Interactive Expressive ECA System.* In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '09, pages 1399–1400, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[Nishimura 1999] Hiroshi Nishimura, Kazuo Hashikawa, Katsumi Doi, Takako Iwaki, Yoshiyuki Watanabe, Hideo Kusuoka, Tsunehiko Nishimura and Takeshi Kubo. *Sign language 'heard' in the auditory cortex.* Nature, vol. 397, no. 6715, pages 116–116, January 1999.

[Nishio 2010] Rie Nishio, S. Hong, S. Konig, Reiner Konrad, Gabriele Langer, Thomas Hanke and Christian Rathmann. *Elicitation methods in the DGS (German Sign Language) corpus project.* In Workshop on the Representation and Processing of Signed Languages, LREC, pages 178–185, 2010.

[Ong 2005] S.C.W. Ong and S. Ranganath. *Automatic sign language analysis: a survey and the future beyond lexical meaning.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 6, pages 873 – 891, June 2005.

[Padden 2003] Carol Padden and Darline Clark Gunsauls. *How the Alphabet Came to Be Used in a Sign Language.* vol. 4, no. 1, pages 10–33, 2003. Volume 4, Number 1, Fall 2003 .

[Padden 2010a] C. Padden, I. Meir, M. Aronoff, W. Sandler and D. Brentari. *The grammar of space in two new sign languages.* In Sign Languages, Cambridge Language Surveys. Cambridge University Press, 2010.

[Padden 2010b] Carol Padden. *Sign Language Geography.* In Gau Mathur and Donna Jo Napoli (eds.), Deaf around the World. Oxford University Press, November 2010.

[Parks 2010] Elizabeth Parks and Jason Parks. *A Sociolinguistic Profile of the Peruvian Deaf Community.* Sign Language Studies, vol. 10, no. 4, pages 409–441, January 2010.

[Partington 1998] Alan Partington. *Patterns and meanings: Using corpora for english language research and teaching.* John Benjamins Publishing, January 1998.

[Pierce 1995] Benjamin C. Pierce. *Foundational Calculi for Programming Languages.* In in the CRC Handbook of Computer Science and Engineering. Available electronically, 1995.

[Pitsikalis 2011] Vassilis Pitsikalis, S. Theodorakis, C. Vogler and P. Maragos. *Advances in phonetics-based sub-unit modeling for transcription alignment and sign language*

recognition. In 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 1–6, June 2011.

[Pizzuto 2001] Elena Pizzuto and Paola Pietrandrea. *The notation of signed texts: Open questions and indications for further research*. Sign Language & Linguistics, vol. 4, no. 1-1, pages 29–45, January 2001.

[Prillwitz 1989] S. Prillwitz, R. Leven, H. Zienert, T. Hanke and J. Henning. *Hamnosys. version 2.0. hamburg notation system for sign language an introductory guide*. Hamburg Signum, Hamburg, Germany, 1989.

[Pullum 2005] Geoffrey K. Pullum and Barbara C. Scholz. *Contrasting applications of logic in natural language syntactic description*. In Logic, Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress, pages 481–503. King's College Publications, 2005.

[Rehg 1994] James M. Rehg and Takeo Kanade. *Visual tracking of high DOF articulated structures: An application to human hand tracking*. In Jan-Olof Eklundh (ed.), Computer Vision — ECCV '94, number 801 de Lecture Notes in Computer Science, pages 35–46. Springer Berlin Heidelberg, 1994.

[Rezzoug 2006] N. Rezzoug, P. Gorce, A. Heloir, S. Gibet, N. Courty, J.F. Kamp, F. Multon and C. Pelachaud. *Virtual humanoids endowed with expressive communication gestures : the HuGEx project*. In IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06, volume 5, pages 4445–4450, October 2006.

[Roth 1993] R. Roth, J. Baker, J. Baker, L. Gillick, M. Hunt, Y. Ito, S. Lowe, J. Orloff, B. Peskin and F. Scattone. *Large vocabulary continuous speech recognition of Wall Street Journal data*. In , 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993. ICASSP-93, volume 2, pages 640–643 vol.2, 1993.

[Sandler 1989] Wendy Sandler. *Phonological Representation of the Sign: Linearity and Nonlinearity in American Sign Language*. Walter de Gruyter, January 1989.

[Sandler 2006] Wendy Sandler and Diane Lillo-Martin. *Sign language and linguistic universals | sign language*. Cambridge University Press, UK, February 2006.

[Saussure 1916] Ferdinand de Saussure, Charles Bally, Albert Sechehaye and Albert Riedlinger. *Cours de linguistique générale*. Payot, Lausanne; Paris, 1916.

[Schembri 2010] Adam Schembri and Onno Crasborn. *Issues in creating annotation standards for sign language description*. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC), pages 212–216, Valletta, Malta, 2010.

[Schembri 2013] Adam Schembri, Jordan Fenlon, Ramas Rentelis, Sally Reynolds and Kearsy Cormier. *Building the British Sign Language Corpus.* Language Documentation and Conservation, vol. 7, pages 136–154, October 2013. National Foreign Language Resource Center.

[Segouat 2009] Jérémie Segouat. *A Study of Sign Language Coarticulation.* SIGACCESS Access. Comput., no. 93, pages 31–38, January 2009.

[Sématos 2013] Sématos. *French Sign Language dictionary.* Online at http://www.sematos.eu/lsf.html, 2013.

[Senghas 2004] Ann Senghas, Sotaro Kita and Asli Ozyurek. *Children Creating Core Properties of Language: Evidence from an Emerging Sign Language in Nicaragua.* Science, vol. 305, no. 5691, pages 1779–1782, September 2004.

[Slobin 2006] D. I Slobin. *Review of S. Liddell, Grammar, gestures, and meaning in American Sign Language.* Language, vol. 82, pages 176–179, 2006.

[Solomon 2010] Andrea Solomon. *Cultural and Sociolinguistic Features of the Black Deaf Community.* Dietrich College Honors Theses, Carnegie Mellon University Press, April 2010.

[Sperling 1986] George Sperling, Michael Landy, Yoav Cohen and M. Pavel. *Intelligible encoding of ASL image sequences at extremely low information rates.* In Papers from the second workshop Vol. 13 on Human and Machine Vision II, pages 256–312, San Diego, CA, USA, 1986. Academic Press Professional, Inc.

[Stamp 2004] Mark Stamp. *A revealing introduction to hidden Markov models.* Department of Computer Science San Jose State University, 2004.

[Starner 1995] Thad Starner. *Visual recognition of American sign language using hidden Markov models.* Thesis, Massachusetts Institute of Technology, 1995.

[Starner 1998] T. Starner, J. Weaver and A. Pentland. *Real-time American sign language recognition using desk and wearable computer based video.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pages 1371–1375, December 1998.

[Stefanov 2013] Kalin Stefanov and Jonas Beskow. *A Kinect Corpus of Swedish Sign Language Signs.* In Proceedings of the 2013 Workshop on Multimodal Corpora: Beyond Audio and Video, 2013.

[Stokoe 1960] William C. Stokoe. *Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf.* Journal of Deaf Studies and Deaf Education, vol. 10, pages 3–37, 1960.

[Stokoe 1976] William C. Stokoe, Dorothy C. Casterline and Carl G. Croneberg. *A dictionary of american sign language on linguistic principles.* Linstok Press Silver Spring, 1976.

[Sutton-Spence 1999a] Rachel Sutton-Spence. *The influence of English on British Sign Language*. International Journal of Bilingualism, vol. 3, no. 4, pages 363–394, December 1999.

[Sutton-Spence 1999b] Rachel Sutton-Spence and Bencie Woll. *The linguistics of british sign language: An introduction*. Cambridge University Press, March 1999.

[Sutton 2000] Valerie Sutton. *Sign writing*. Deaf Action Committee (DAC), 2000.

[Sutton 2009] Valerie Sutton. *SignWriting basics instruction manual*. SignWriting Press, first edition edition edition, 2009.

[Sutton 2010] Valerie Sutton. *SignPuddle Online*, 2010.

[Takahashi 1991] Tomoichi Takahashi and Fumio Kishino. *Hand Gesture Coding Based on Experiments Using a Hand Gesture Interface Device*. SIGCHI Bull., vol. 23, no. 2, pages 67–74, March 1991.

[Tebbe-Grossman 2008] Jennifer Tebbe-Grossman. *Writing Deafness: The Hearing Line in Nineteenth-Century American Literature by Christopher Krentz*. The Journal of American Culture, vol. 31, no. 2, pages 267–269, June 2008.

[Timmermans 2005] Nina Timmermans. The status of sign language in europe, pages 46–48. Council of Europe Publishing, 2005.

[Tonkin 1997] Humphrey Tonkin. *Esperanto, Interlinguistics, and Planned Language*. University Press of America, Lanham, Md. : Rotterdam, Netherlands ; West Hartford, CT, 5 edition edition, September 1997.

[Valli 2000] Clayton Valli and Ceil Lucas. *Linguistics of american sign language text, 3rd edition: An introduction*. Gallaudet University Press, 2000.

[van der Hulst 2010] Harry van der Hulst and Rachel Channon. *Notation Systems*. In Diane Brentari (ed.), Sign Languages, Cambridge Language Surveys, pages 151–172. Cambridge University Press, May 2010.

[Venkatagiri 2014] Horabail Venkatagiri. *Lecture Notes on American Sign Language*. Iowa State University. Online at http://www.public.iastate.edu/ giri/cmdis286xw/graphics, 2014.

[Vercher 2006] Jean-Louis Vercher. *Perception and Synthesis of Biologically Plausible Motion: From Human Physiology to Virtual Reality*. In Sylvie Gibet, Nicolas Courty and Jean-François Kamp (eds.), Gesture in Human-Computer Interaction and Simulation, number 3881 de Lecture Notes in Computer Science, pages 1–12. Springer Berlin Heidelberg, 2006.

[Verlinden 2002] Margriet Verlinden, Corrie Tijsseling and Han Frowein. *A Signing Avatar on the WWW*. In Ipke Wachsmuth and Timo Sowa (eds.), Gesture and Sign Language in Human-Computer Interaction, number 2298 de Lecture Notes in Computer Science, pages 169–172. Springer Berlin Heidelberg, January 2002.

[Vigliocco 2014] Gabriella Vigliocco, Pamela Perniss and David Vinson. *Language as a multimodal phenomenon: implications for language learning, processing and evolution*. Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 369, no. 1651, September 2014.

[Vogler 1997] C. Vogler and D. Metaxas. *Adapting hidden Markov models for ASL recognition by using three-dimensional computer vision methods*. In , 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation, volume 1, pages 156–161 vol.1, October 1997.

[Vogler 2001] Christian Vogler and Dimitris Metaxas. *A Framework for Recognizing the Simultaneous Aspects of American Sign Language*. Computer Vision and Image Understanding, vol. 81, no. 3, pages 358–384, March 2001.

[Vogler 2004] Christian Vogler and Dimitris Metaxas. *Handshapes and Movements: Multiple-Channel American Sign Language Recognition*. In Antonio Camurri and Gualtiero Volpe (eds.), Gesture-Based Communication in Human-Computer Interaction, number 2915 de Lecture Notes in Computer Science, pages 247–258. Springer Berlin Heidelberg, 2004.

[Vogler 2007] Christian Vogler and Siome Goldenstein. *Facial movement analysis in ASL*. Universal Access in the Information Society, vol. 6, no. 4, pages 363–374, November 2007.

[Von Agris 2008] U. Von Agris, M. Knorr and K.-F. Kraiss. *The significance of facial features for automatic sign language recognition*. In 8th IEEE International Conference on Automatic Face Gesture Recognition, 2008. FG '08, pages 1–6, September 2008.

[Waldron 1995] M.B. Waldron and Soowon Kim. *Isolated ASL sign recognition system for deaf persons*. IEEE Transactions on Rehabilitation Engineering, vol. 3, no. 3, pages 261–271, September 1995.

[Wang 2002] Chunli Wang, Wen Gao and Shiguang Shan. *An approach based on phonemes to large vocabulary Chinese sign language recognition*. In Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 2002. Proceedings, pages 411–416, 2002.

[Wang 2012] Haijing Wang, Alexandra Stefan, Sajjad Moradi, Vassilis Athitsos, Carol Neidle and Farhad Kamangar. *A System for Large Vocabulary Sign Search*. In Kiriakos N. Kutulakos (ed.), Trends and Topics in Computer Vision, number 6553 de Lecture Notes in Computer Science, pages 342–353. Springer Berlin Heidelberg, 2012.

[Warren 2004] Martin Warren. *A Corpus-driven Analysis of the Use of Intonation to Assert Dominance and Control*. Language and Computers, vol. 52, no. 1, pages 21–33, October 2004.

[WebSourd 2015] WebSourd. *Signeur Virtuel* 3DSigner. Online at http://www.3dsigner.fr/, January 2015.

[Wilbers 1987] Stephen Wilbers. *The Case for Recognizing American Sign Language*. College Board Review, no. 145, pages 4–8, Fall 1987.

[Wilbur 2009] Ronnie B. Wilbur. *Effects of Varying Rate of Signing on ASL Manual Signs and Nonmanual Markers*. Language and speech, vol. 52, no. 2-3, pages 245–285, 2009.

[Wilson 1996] A.D. Wilson, A.F. Bobick and J. Cassell. *Recovering the temporal structure of natural gesture*. In , Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996, pages 66–71, 1996.

[Wing 1990] Jeannette M. Wing. *A Specifier's Introduction to Formal Methods*. Computer, vol. 23, no. 9, pages 8–23, September 1990.

[Winskel 1995] Glynn Winskel and Mogens Nielsen. *Handbook of logic in computer science*, volume 4. Oxford University Press, Oxford, UK, 1995.

[Winstanley 1991] A.C. Winstanley and D.W. Bustard. *EXPOSE: an animation tool for process-oriented specifications*. Software Engineering Journal, vol. 6, no. 6, pages 463–475, November 1991.

[Wittenburg 2006] Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann and Han Sloetjes. *ELAN : a professional framework for multimodality research*. In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006), pages 1556–1559, 2006.

[Woodcock 1996] Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[Woodcock 2009] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui and John Fitzgerald. *Formal Methods: Practice and Experience*. ACM Comput. Surv., vol. 41, no. 4, pages 19:1–19:36, October 2009.

[Woodward 1976] James C. Woodward. *Signs of Change: Historical Variation in American Sign Language*. Sign Language Studies 10, January 1976.

[Wu 1999] Ying Wu and Thomas S. Huang. *Vision-Based Gesture Recognition: A Review*. In Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction, GW '99, pages 103–115, London, UK, UK, 1999. Springer-Verlag.

[Xu 1993] Jun Xu, Yoshinao Aoki and Zhong Zheng. *A method for synthesizing animation to transmit sign language by intelligent communication*. Electronics and Communications in Japan (Part III: Fundamental Electronic Science), vol. 76, no. 2, pages 108–117, January 1993.

[Yang 2006] Ruiduo Yang, Sudeep Sarkar, Barbara Loeding and Arthur Karshmer. *Efficient Generation of Large Amounts of Training Data for Sign Language Recognition: A Semi-automatic Tool.* In Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler and Arthur I. Karshmer (eds.), Computers Helping People with Special Needs, number 4061 de Lecture Notes in Computer Science, pages 635–642. Springer Berlin Heidelberg, 2006.

[Yang 2009] Hee-Deok Yang, S. Sclaroff and Seong-Whan Lee. *Sign Language Spotting with a Threshold Model Based on Conditional Random Fields.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 7, pages 1264–1277, July 2009.

[Yoo 2008] Junbeom Yoo, Sungdeok Cha and Eunkyoung Jee. *A Verification Framework for FBD Based Software in Nuclear Power Plants.* In Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific, pages 385–392, December 2008.

[Zafrulla 2011] Zahoor Zafrulla, Helene Brashear, Thad Starner, Harley Hamilton and Peter Presti. *American Sign Language Recognition with the Kinect.* In Proceedings of the 13th International Conference on Multimodal Interfaces, ICMI '11, pages 279–286, New York, NY, USA, 2011. ACM.

[Zahedi 2005] Morteza Zahedi, Daniel Keysers and Hermann Ney. *Appearance-Based Recognition of Words in American Sign Language.* In Proceedings of the Second Iberian Conference on Pattern Recognition and Image Analysis - Volume Part I, IbPRIA'05, pages 511–519, Estoril, Portugal, 2005. Springer-Verlag.

[Zahedi 2006] Morteza Zahedi, Philippe Dreuw, David Rybach, Thomas Deselaers and Hermann Ney. *Continuous sign language recognition-approaches from speech recognition and available data resources.* In LREC Wkshp: Represent. and Process. of Sign Languages, pages 21–24, 2006.

[Zambach 2010] Sine Zambach and Jens Ulrik Hansen. *Logical Knowledge Representation of Regulatory Relations in Biomedical Pathways.* In Sami Khuri, Lenka Lhotská and Nadia Pisanti (eds.), Information Technology in Bio- and Medical Informatics, ITBAM 2010, number 6266 de Lecture Notes in Computer Science, pages 186–200. Springer Berlin Heidelberg, 2010.

[Zhao 2000] Liwei Zhao, Karin Kipper, William Schuler, Christian Vogler, Norman Badler and Martha Palmer. *A Machine Translation System from English to American Sign Language.* In John S. White (ed.), Envisioning Machine Translation in the Information Future, number 1934 de Lecture Notes in Computer Science, pages 54–67. Springer Berlin Heidelberg, 2000.