GENERATING AMERICAN SIGN LANGUAGE CLASSIFIER PREDICATES FOR ENGLISH-TO-ASL MACHINE TRANSLATION

Matt Huenerfauth

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2006

Acknowledgements

I want to thank my advisors Dr. Mitch Marcus and Dr. Martha Palmer for being so generous with their time and advice during the past four years. I am grateful for the trust they placed in me when I came to them wanting to pursue this new avenue of research and for the support they gave me along the way. They have encouraged me to explore new research directions, supported my study of ASL, helped me to establish ties to the research community, and given me broad autonomy in directing this research project. They have inspired me both academically and professionally, and I know that I have learned the skills I will need to be a successful academic researcher through their advice and example. I feel doubly fortunate to have had them as my advisors.

I would also like to thank Dr. Norman Badler for his advice and support – particularly with the animation aspects of this research. I'm grateful for his willingness to make the resources of the Center for Human Modeling and Simulation available to this project. Jan Allbeck, Erdan Gu, and Liming Zhao – Ph.D. students from the Center – were actively involved in this project, and some sections of this dissertation describe joint work with them: Motion-Planning for the Hand and Arm Animation (Chapter 8), Motion-Planning for Eye-Gaze and Eye-Brow Raising (Chapter 8), Displaying the Animation (Chapter 8), Collection of ASL Motion-Capture Data (Chapter 9), and Creating Animations from the Motion-Capture Data (Chapter 9). Research into ASL generation and machine translation requires this kind of successful collaboration between natural language processing and animation/simulation researchers, and I'm grateful for the technological expertise and insights that they have brought to this project.

I would also like to thank Katherine Feldmann for contributing her native ASL signing skills, methodological recommendations, and contacts within the Deaf community to help make the evaluation portion of this project a success. I would also like to thank all of the study participants, who offered insightful feedback about the performance of our prototype ASL generation system.

I would also like to acknowledge the following organizations and individuals for supplying resources to the Center for Human Modeling and Simulation that were used in this project: MotionBuilder animation software from Autodesk, Inc.; Jack digital human modeling and ergonomics software from UGS; the Greta animated human head from Catherine Pelachaud; and the LiveActor motion-capture studio made possible through grants and support furnished by the National Science Foundation, University of Pennsylvania's School of Engineering and Applied Science, Ascension Technology Corporation, and EON Reality.

I am grateful for several forms of funding this project has received from the National Science Foundation. An NSF Graduate Research Fellowship has supported the last three years of my doctoral work, and an NSF Small Grant for Exploratory Research (SGER) has supported the evaluation study of our prototype generation system – NSF Award #0520798 "SGER: Generating Animations of American Sign Language Classifier Predicates" (Universal Access Program).

Finally, I would like to thank my family for their love and support throughout this Ph.D. experience – and throughout everything else. In a million ways, both big and small, they have gotten me to this point, and I’m really grateful. I remember my mom teaching me reading and math when I was a child, my dad sitting down with me to figure out how to program a Commodore 64 computer, my sister helping me glue together a science project the night before it was due, my parents working to send their kids to good schools, the whole family moving me in and out of countless dorm rooms, and no one ever questioning me why I needed to go for that next degree or chase that next dream. While the ways in which they have helped me have changed throughout the years, one thing has stayed the same: they’ve always been there when I’ve needed them, and nothing was ever too much to ask.

ABSTRACT

GENERATING AMERICAN SIGN LANGUAGE CLASSIFIER PREDICATES

FOR ENGLISH-TO-ASL MACHINE TRANSLATION

Matt Huenerfauth

Mitch Marcus and Martha Palmer

A majority of deaf 18-year-olds in the United States have an English reading level below that of a typical 10-year-old student, and so machine translation (MT) software that could translate English text into American Sign Language (ASL) animations could significantly improve these individuals' access to information, communication, and services. Previous English-to-ASL MT projects have made limited progress by restricting their output to subsets of ASL phenomena – thus avoiding important linguistic and animation issues. None of these systems have shown how to generate classifier predicates (CPs), a phenomenon in which signers use special hand movements to indicate the location and movement of invisible objects (representing entities under discussion) in space around their bodies. CPs are frequent in ASL and are necessary for conveying many concepts.

This project has created an English-to-ASL MT design capable of producing classifier predicates. The classifier predicate generator inside this design has a planning-based architecture that uses a 3D "visualization" model of the arrangement of objects in a scene discussed by the English input text. This generator would be one pathway in a multi-path English-to-ASL MT design; a separate processing pathway would be used to generate classifier predicates, to generate other ASL sentences, and to generate animations of Signed English (if the system lacked lexical resources for some input).

Instead of representing the ASL animation as a string (of individual signs to perform), this system encodes the multimodal language signal as multiple channels that are hierarchically structured and coordinated over time. While this design feature and others have been prompted by the unique requirements of generating a sign language, these technologies have applications for the machine translation of written languages, the representation of other multimodal language signals, and the production of meaningful gestures by other animated virtual human characters.

To evaluate the functionality and scalability of the most novel portion of this English-to-ASL MT design, this project has implemented a prototype-version of the planning-based classifier predicate generator. The classifier predicate animations produced by the system have been shown to native ASL signers to evaluate the output.

Table of Contents

List of Tables

List of Illustrations

Chapter 1:
Introduction and Overview

This chapter will give a brief overview of the motivations, applications, and design goals of this English to American Sign Language (ASL) machine translation project. This chapter will discuss some key properties about the system and its design that will be demonstrated throughout this dissertation, and it will highlight some of the implementation and evaluation accomplishments of the project. The chapter will conclude by describing the structure of this document in a chapter-by-chapter manner.

Overview of this Project

Many deaf adults have low levels of English literacy because of a lack of exposure to accessible language input during the critical language-acquisition years of childhood. Unfortunately, many accessibility tools for the deaf, like television closed-captioning or teletype telephones, assume the user has strong English literacy skills. Many deaf adults with English reading difficulty are fluent in American Sign Language (ASL), a natural language with a linguistic structure distinct from English. Therefore, English-to-ASL machine translation (MT) software could make information and services accessible to deaf users when English text captioning is too complex or an interpreter is unavailable.

English-to-ASL translation is at least as complex as translation between pairs of spoken languages, and in fact, ASL's visual modality introduces unique complexities into the translation problem. For example, the space around the signer can be used for many communicative purposes, including the production of constructions called "classifier predicates." These complex hand movements trace contours, identify locations, or draw paths through the space in front of the signer that topologically correspond to the shape, location, or movement of real world entities in a three-dimensional scene.

Some previous MT projects have made initial efforts at translating English text into ASL animation, but none have proposed a practical design for generating classifier predicates. These phenomena are a necessary and frequent part of ASL signing, and so previous systems can only produce ASL with limited fluency. The English sentences these systems cannot translate (involving spatial concepts) would be important to translate for some accessibility applications.

The goal of this project is to design a classifier predicate generator that can be incorporated into a complete English-to-ASL MT design. There are several major components to the overall architectural design described throughout this dissertation:

- A planning-based classifier predicate generator that uses scene-visualization software to produce a 3D model of the real-world scene being discussed in an English input text.

- An overall multi-path English-to-ASL MT design that includes separate translation pathways for producing classifier predicates (using the design above) and other forms of ASL signing.

- A multichannel ASL representation that serves the output of the linguistic generation portion of the system (and as the input to the animation portion).
- A set of ASL linguistic models that are accessed and modified during the classifier predicate generation (some of which can also be used during the generation of non-classifier-predicate output).

Key Properties of the System

This dissertation will demonstrate several important properties of the classifier predicate generator and its design:

- The generator design is ***well-specified*** and ***coherent*** – that is, the design can actually be used to build a classifier predicate generator.
- The generator is ***useful for MT*** – that is, the classifier predicate generator could be used as part of a complete English-to-ASL MT design.
- The generator is ***functional*** – that is, it can successfully produce animations of ASL classifier predicates understandable to native ASL signers.
- The generator is ***scalable*** – that is, we can estimate the effort required to extend the coverage of the system and demonstrate how adding new linguistic resources to the system does not interfere with its previous functionality.
- The generator is ***robust*** – that is, an English-to-ASL MT system including this classifier predicate generator would gracefully degrade in performance if it

encounters an input text that it cannot translate successfully (e.g. if some part of the input text is out-of-vocabulary for the classifier predicate generator).

There are other attractive properties that the system and its design may possess yet which are not critical to demonstrate to justify the major research claims of this thesis.

- The generator design is linguistically interesting and/or suggestive.
- The generator has an elegant object-oriented software implementation.
- The generator design has implications/applications beyond sign language.

While these final three properties have not been specifically addressed by the implementation or evaluation portions of this dissertation project, some of them serve as discussion topics in Chapter 11.

Implementation and Evaluation

While this dissertation presents a design for an overall English-to-ASL MT system, it is the classifier predicate generator which is the focus of this research project. Since there are no other English-to-ASL MT systems that have addressed the generation of these phenomena, it is this component of the design which is most important to demonstrate is coherent, functional, scalable, and robust. To verify some of these properties, it was necessary to build a prototype version of the classifier predicate generator, determine how much development effort went into its implementation, and evaluate its output. While the non-classifier-predicate portions of the English-to-ASL MT design have not been implemented (see discussion of system implementation in Chapter 8), they have been important to specify: without the entire design, we would not be able to demonstrate that the classifier predicate generator could be used within a complete MT system.

Various portions of the English-to-ASL MT design have been implemented as part of this project: a planning-based classifier predicate generator, an initial set of classifier predicate templates, important data types and interfaces used in the system, software implementations of some ASL linguistic models, and a multichannel ASL representation accessed during generation. After a prototype of the system was built, native ASL signers were asked to participate in an evaluation of the system's animation output.

Structure of this Dissertation

Chapters 1, 2, and 3 serve as the introduction and background for this dissertation; they include the motivations, the ASL linguistic foundations, and the literature survey. **Chapter 2** discusses a set of common misconceptions about deafness and literacy, the language status of ASL, the special challenges faced during ASL MT, and the importance of generating classifier predicates. This chapter also provides a review of ASL linguistics and current accessibility tools for the deaf. **Chapter 3** is a survey of previous work on English-to-ASL machine translation. This chapter will highlight four major failures of previous work that will be addressed by the MT design described in this dissertation.

Chapters 4, 5, 6, and 7 describe the design of an English-to-ASL MT system that is capable of producing classifier predicates. **Chapter 4** describes the design of a planning-based classifier predicate generator and the various ASL linguistic models that will be used within this component. The chapter uses the several competing linguistic theories of ASL classifier predicates to organize its discussion of possible implementations of this component. **Chapter 5** explains how a classifier predicate generator could be incorporated into a complete English-to-ASL MT design. This chapter also includes some discussion of how the non-classifier-predicate portions of the design could be implemented (the implementation of these components is outside the scope of this dissertation). **Chapter 6** explains the development of a multichannel ASL representation that has several advantages over string-based encodings of ASL used by previous researchers. The chapter defines the Partition/Constitute Formalism, which underlies this representation. **Chapter 7** describes how the ASL linguistic models used in this system can be built from a small set of linguistic and spatial data types.

Chapter 8 discusses the implementation of the prototype version of the classifier predicate generator. This chapter explains how a subset of the design was chosen to be implemented for this dissertation project. The chapter also gives an extended example of how an English sentence is translated into ASL classifier predicates using the design described in Chapters 4, 6, and 7.

Chapter 9 discusses how to evaluate the prototype classifier predicate generator described in Chapter 8. This chapter discusses how a user-based evaluation study of the prototype generator was conducted. This chapter also describes and interprets the results of this study, and it suggests ways to refine the design of future studies.

Chapter 10 discusses some future work that follows from this dissertation. This chapter suggests modifications to the classifier predicate generator, further development of the English-to-ASL MT system, and the design of future evaluation studies.

Chapter 11 outlines the major the research claims of this dissertation, highlights the contributions of this project, and summarizes how the special requirements of ASL generation are responsible for key elements of the system's design. This chapter also describes how the new MT technology developed for this project has applications beyond the generation of sign languages, and it discusses how ASL research can contribute to the rest of the field of natural language processing.

Chapter 2:
ASL MT Motivations and Misconceptions

There has been limited progress in the development of machine translation software to convert English text into American Sign Language (ASL) animation, despite the benefits such software could have for the majority of deaf Americans who face English literacy challenges (Huenerfauth, 2004c, 2005a, 2005c). This chapter will explore how accessibility technology has been slow to address this literacy issue because of misconceptions surrounding: the rate of written English literacy among the deaf, the lack of an ASL writing system, the natural language status of ASL, the importance of certain ASL phenomena called "classifier predicates," and the suitability of traditional computational linguistic software to the special linguistic properties of ASL.

Misconception: All Deaf Users are Written-English Literate

During the critical language-acquisition years of childhood, a deaf individual requires sufficient exposure to an accessible language input (e.g. a sign language, which is transmitted visually) to enable the acquisition of a first language (in this case, a sign language). From this foundation, the later acquisition of literacy in a spoken/written language like English would be facilitated. Unfortunately, many deaf individuals do not have an opportunity to fully acquire a first language in this manner, and many have below-average levels of literacy in a written language. In fact, studies have shown that the majority of deaf high school graduates in the United States have only a fourth grade English reading level (Holt, 1991). This means that students age 18 and older have a reading level more typical of a 10-year-old student.

The primary means of communication for an estimated one-half million to two million deaf people in the United States is American Sign Language (ASL), a full natural language with a linguistic structure distinct from English (Lane et al., 1996, Neidle et al., 2000, Liddell, 2003a; Mitchell, 2004). Thus, it is possible to have fluency in ASL without literacy in written English. This literacy issue has become more significant in recent decades as new information and communications technologies have arisen that place an even greater premium on written English literacy in modern society.

Deaf Accessibility Tools and English Literacy

Many accessibility 'solutions' for the deaf simply ignore part of the problem – often designers make the assumption that the deaf users of their tools have strong English reading skills. For example, television "closed captioning" converts an audio English signal into visually presented English text on the screen; however, the reading level of this text may be too high for many deaf viewers. While some programming may be accessible with this approach, deaf users may be cut off from important information contained in news broadcasts, educational programming, political debates, and other broadcasts with a more sophisticated level of English language. Communications technologies like teletype telephones (sometimes referred to as telecommunications devices for the deaf or TDDs) similarly assume the user has English literacy. The user is expected to both read and write English text in order to have a conversation. Many software designers incorrectly assume that written English text in a user-interface is always accessible to deaf users. Few software companies have addressed the connection between deafness and literacy, and so few computer user-interfaces make sufficient accommodation for the deaf. Many designers believe that if audio information is also presented as written English text, then the needs of a deaf user are met.

A machine translation system from English text into American Sign Language animations could increase the accessibility of all of these technologies. Instead of presenting written text on a television screen, telephone display, or computer monitor, each could instead display ASL output. An automated English-to-ASL machine translation (MT) system could make information and services accessible when English text captioning is too complex, an English-based user-interface is too difficult to navigate, or when live interpreting services are unavailable. This type of MT software could also be used to build new educational software for deaf children to help them improve their English literacy skills[1]. As part of this dissertation project, an important component for such an English-to-ASL MT system has been developed, specifically a system that can produce animations of complex ASL phenomena called "classifier predicates" (described later in this chapter).

[1] If English-to-ASL machine translation technology were used in English language education software for deaf children, then this technology could actually help to combat the English literacy disparity which originally motivated its existence.

Misconception: We Can Generate ASL Text as Output

ASL is a language without a standard written form. While several ASL writing systems have been proposed (Newkirk, 1987; Sutton, 1998), none have gained significant popularity among ASL signers. Without a community that accepts and develops literacy skills in one of these writing systems, an ASL MT system could not use any of them as output – it must produce an animation of a character performing ASL.

Animated ASL Signing Characters

Research into virtual reality human modeling and animation has reached the point of sophistication where it is now possible to construct a model of the human form which is articulate and responsive enough to perform American Sign Language. The level of quality of such human avatar animations has increased such that human signers of ASL can now view the onscreen animations and successfully interpret the movements of the avatar to understand its meaning (Wideman & Sims, 1998). However, just because graphics researchers know how to move the character, this doesn't mean that we have ASL generation software. Given an English text or some semantic input representation, a computational linguistic component would need to tell the animated character what to do (assuming the correct instruction set for the interface between the linguistics and the animation components has been determined). An English-to-ASL translation system would need to take an English text input and produce an ASL output "script" for an animated character to follow. Since ASL has no writing system, the best script specification to use as a linguistics-animation interface is an open area of research.

There are many animated human characters that have already been built by graphics animation researchers, and some researchers have even developed characters that have been specifically designed to perform sign-language movements (Elliot et al., 2004; Wideman & Sims, 1998). Thus, natural language processing researchers can now focus on the linguistic portion of the English-to-ASL animation task and rely on computer graphics researchers to help supply the animation technology to produce the final output animation based on the “script” produced by the linguistic portions of the MT system.

Building ASL Corpora

The lack of an ASL writing system has also made it difficult and expensive to collect large corpora of ASL with sufficient detail for computational linguistic research. Unlike users of written languages who produce large collections of text (in news reports, on the Internet, etc.), ASL signers do not naturally produce written “texts” of ASL. Corpora builders cannot harvest pre-existing text resources when producing an ASL corpus for linguistic or natural language processing research. Instead, researchers videotape signers performing ASL, and this video is carefully annotated by linguists to produce a symbolic record of the movements made during the performance (Neidle et al., 2001).

The collection and annotation of ASL performances is a challenging and time-consuming process, and so the rate of growth of ASL corpora has been much slower than that for written languages. The difficulty in quickly building large ASL corpora has prevented many of the popular stochastic machine translation approaches from being applied to ASL (since most would require large amounts of parallel English-ASL language data to train a machine learning MT algorithm). For this reason, the ASL MT systems discussed in the literature survey (Chapter 3) and the English-to-ASL MT design described in this dissertation use non-statistical machine translation technologies.

Misconception: ASL is Just Manually Performed English

Even when researchers understand that presenting English text is not a complete accessibility solution for deaf users, confusion within the natural language processing research community over the language status of ASL has delayed the creation of MT technology. Many researchers have assumed that the reason why many deaf people have difficulty reading English is that it is presented in the form of words written in Roman alphabet letters. Under this assumption, if we were to replace every word of an English sentence with a corresponding ASL sign (the assumption is also made that such a correspondence always exists), then deaf users would be able to understand the text.

There is a common misconception that English and ASL have the same linguistic structure – that one language is merely a direct encoding of the other. In fact, the word order, linguistic structure, and vocabulary differences between English and ASL are comparable to those between many pairs of written languages. And while there are some signing communication systems that use English structure, these are often limited to use in English classrooms for the deaf. In most cases, presentation of ASL signs in English word order (and without the accompanying ASL linguistic information contained in facial expressions, eye gaze, etc.) will not be understandable to a deaf user.

ASL vs. Signed English

It is important to distinguish the following methods of manual communication:

- **American Sign Language (ASL)** is the primary means of communication for approximately one half million deaf people in the United States (Mitchell, 2004). It has been well-established that ASL is a full natural language, despite differences between it and spoken languages (Neidle et al., 2000; Liddell, 2003a). The differences between English and ASL are significant enough that fluency in one language does not imply fluency in the other.

- **Signed English (SE)** is another form of manual signing communication that is distinct from ASL and is not a full natural language. There are several different styles of SE communication, but all of them encode an English sentence into a set of signs performed by the signer's hands. SE uses many of the same signs as ASL (and some additional signs of its own). SE retains English sentence structure and word order, and it is most commonly used in educational settings for deaf students who are learning English. The structure, word order, and linguistic phenomena of ASL are often different than those of an SE performance.

- **Fingerspelling** is a method of communicating the letters of the English alphabet using special handshapes to spell words during signing. Signers typically do not use fingerspelling to convey arbitrary words during ASL; it is usually reserved for titles, proper names, and other specific situations. In an English-to-ASL interpretation/translation context, fingerspelling can be more common; an interpreter will often use fingerspelling to directly mimic terminology or specialized vocabulary being used in the original English language signal.

A human ASL interpreter may need to use multiple forms of manual communication when translating an English language signal for a client who is deaf. While many deaf clients prefer ASL signing, sometimes interpreters will perform signing that varies on a spectrum between ASL and Signed English. Occasionally, the client may request English-like signing, but other times source-language influences or the cognitive challenge of simultaneous interpretation will lead the interpreter to inadvertently produce English-like ASL signing. Many deaf clients will accept ASL that contains a small amount of English-like influence; however, the extensive use of English-like structure will lead to a non-fluent ASL performance that is difficult to understand.

Confusion over the linguistic status of ASL (and its relationship to Signed English) has led some computational linguistic researchers to produce "translation" systems that produce Signed English and not ASL. (See Chapter 3 for a discussion of these systems.) Since the animation output of such systems contains entirely English-like structure (and not ASL), it is of limited usefulness to deaf adults with low English-literacy skills (the target users of ASL MT software).

Misconception: ASL Can Easily Use Written-Language MT Technology

The structure of ASL is quite different than most written/spoken languages, and its visual modality allows it to use phenomena not seen in these languages (Neidle et al. 2000; Liddell 2003a). In addition to using hands, facial expression, eye gaze, head tilt, and body posture to convey meaning, an ASL signer can use the surrounding space for communicative purposes. It is this use of space around the body which makes it particularly difficult to apply traditional MT technology to ASL – we need representations of space that allow us to successfully generate these spatial phenomena.

This section will summarize some important ASL linguistic terminology, and it will particularly focus on the way in which signers can use the space around them to convey meaning. Traditional written-language MT software designed to generate strings of text would have difficulty addressing many of these complex spatial linguistic phenomena.

The ASL Signing Space

The **Signing Space** is the area of space in front of the signer's torso where the majority of classifier predicates and other ASL signs are performed. During an ASL conversation, signers can associate locations in space around their body with people, objects, or concepts under discussion. Sometimes the way in which objects are arranged in space is meant to topologically indicate the 3D layout of those objects in some scene under discussion. When the signing space is used in this manner, it is referred to as **Depicting Space** (Liddell, 2003a). Other times, the locations associated with objects in space is not meant to indicate how the objects are arranged in a 3D scene. In this case, we say that the space around the signer is being used as a **Token Space** (Liddel, 2003a).[2]

[2] These two "Space" terms were used by Liddell (2003a) in his "blended spaces" model of ASL spatial phenomena. Liddell also used the term Surrogate Space to refer to the space around signers when they are using their own body as a referent during a classifier predicate construction (other researchers refer to these phenomena as Body Classifiers or Narrative Role Shift). During the use of surrogate space, the signer is able to take the place of some character in a scene or narrative currently being described. There are additional complexities that arise during these types of expressions (in the use of emotive facial expressions or positioning of other characters in the scene relative to the signer's own body) which are beyond the scope of the English-to-ASL MT design presented in this dissertation.

During an ASL performance, signers can switch between two **Subsystems** of signing. Both of these subsystems of signing can use the space around the signer in a linguistically meaningful manner; however, they use the space in different ways. One subsystem primarily uses the Signing Space as a Token Space, and the other subsystem primarily uses the Signing Space as a Depicting Space.

There are exceptions to this generalization. Occasionally, a signer can conceptualize the Signing Space as both a Depicting Space and a Token Space at the same time. Other times, the use of the Signing Space as a Depicting Space could affect the locations chosen for objects during later use of the Signing Space as a Token Space.[3]

The Lexical Signing (LS) Subsystem

A signer constructs most ASL sentences by syntactically combining individual ASL lexical items into complete sentences. In addition to assembling lexical items in the proper sequence, signers are required to add grammatically meaningful facial expressions and other non-manual signals (NMS) to the performance. The exact performance of each lexical item (each sign) may also need to be modified to accommodate syntactic, morphological, or phonological requirements of ASL. This form of signing, since it is fundamentally the generation of sentences based on individual lexical signs, will be called **Lexical Signing** (or LS) in this document. The major indicator that a signer is currently using the **Lexical Signing (LS) Subsystem** of American Sign Language is that their performance does not include any classifier predicates (which will be discussed later

[3] These phenomena are beyond the scope of the English-to-ASL MT design presented in this dissertation, but with some modification, they could be handled within the MT framework presented in later chapters.

in this chapter). Figure 3(a) on page 39 contains an example of an ASL LS sentence; this sentence will be described and discussed in Chapter 3.

During LS signing, entities under discussion (whether a concrete person/object or an abstract concept) can be associated with a particular 3D location in the signing space. During LS signing, the Signing Space is primarily used as a Token Space. This means that the locations chosen for entities during LS signing are not topologically meaningful; that is, one entity being positioned to the left of another in the signing space does not indicate the entity is to the left of the other in the real world. These locations chosen for entities can be used during pronominal reference; after an object is associated with a location in space, subsequent pronominal reference to this entity can be made by pointing to this location (Neidle et al., 2000). Some ASL verb signs, called "agreement verbs," move toward or away from these locations in space to indicate (or show agreement with) their arguments (Padden, 1988; Neidle et al., 2000). During an LS sentence, a signer can also use their head-tilt and eye-gaze to indicate locations in space that are associated with the subject and object of the verb being performed (Neidle et al., 2000).

While the signing space is typically used as a Token Space during LS sentence, there are exceptions. Some ASL verbs have the ability to modify their 3D movement path to indicate how an object moves through space. These "spatial" or "depicting" verbs (Padden, 1988; Liddell, 2003a) are different than the more common "agreement" verbs described above. While agreement verbs can move to indicate a location associated with its subject (and sometimes also its object) in the Token Space, depicting verbs can modify their movement path to indicate some 3D spatial information about the way in which the action described by the verb was performed. For instance, during the performance of the ASL verb MOVE, a signer can actually indicate a path along which an object was moved (including topologically meaningful starting and ending location). The majority of ASL verbs do not indicate this form of 3D path/movement information (Liddell, 2003a). Since the focus of this dissertation is actually the generation of ASL sentences in the Classifier Predicate Subsystem of the language (see below), we will simplify our discussion for the remainder of this dissertation by continuing with the generalization that ASL LS sentences primarily use the signing space as a Token Space.

The Classifier Predicate (CP) Subsystem

Other ASL phenomena do make use of the space around the signer in a topologically meaningful way; these constructions are called **Classifier Predicates** (or CPs). Classifier predicates are produced by the **Classifier Predicate (CP) Subsystem** of ASL.[4] During a CP, the signers' hands represent an entity in space in front of them, and they position, move, trace, or re-orient this imaginary object to indicate the location, movement, shape, or other properties of some corresponding real world entity under discussion. A CP consists of two simultaneous components: (1) the hand in a semantically meaningful handshape and (2) a 3D movement path that the hand travels through space in front of the signer. To produce a complete ASL sentence, a CP will typically follow a noun phrase that indicates which object in the conversation is being described by the CP.

[4] This CP vs. LS subsystem terminology is unique to this thesis; previous ASL analyses have not proposed as succinct a nomenclature for distinguishing these forms of ASL signing. Liddell (2003a) would describe the LS subsystem as ASL signing that is optionally making use of "token blends" and the CP subsystem as ASL signing that is using "depicting blends," "surrogate blends," or both. The LS/CP acronyms are more convenient. Further, the term subsystem is inspired by a functional linguistics conceptualization of the decision an ASL signer must make as to which form of signing to produce during generation. Talmy (2003) used the term "system" when discussing ASL classifier predicates, but it was not clear whether he used the term to refer to a different generation process or to simply refer to a subset of the ASL lexicon.

A classifier predicate is created by first selecting one of a closed set of handshapes based on the characteristics of the entity in the noun phrase (whether it be a vehicle, upright animate figure, squat four-legged object, etc.) and what aspect of the entity the signer wishes to discuss (its surface, size, position, motion, etc). The signer then produces a three-dimensional movement for the hand which communicates a contour, a position in the space around the signer, a motion through 3D-space, a physical/abstract dimension, and/or some other property of the object which needs to be communicated. Classifier predicates are therefore ideal for describing scenes, articulation of tools, movements, sizes, and other information of a visual/spatial or scene/process nature.

For example, to express "the car parked between the cat and the house," the signer could use a combination of three classifier predicates (Figure 1 contains images of the classifier handshapes used in this example and others used later in this dissertation):

- The non-dominant hand in a "Spread C" handshape would indicate a location in space where a miniature invisible house could be envisioned. The signer's eye gaze would aim at the location assigned to the 'house.'
- The dominant hand in a "Hooked V" handshape would indicate a location in space where a miniature invisible cat could be envisioned. The signer's eye gaze would aim at the location assigned to the 'cat.'
- The dominant hand in a "Number 3" handshape would trace a path in space corresponding to a car driving and stopping in between the 'house' and 'cat' locations in space. As the car nears the end of its motion path, the open palm of

the non-dominant hand (in a "Flat B" handshape) would form a platform on which the car parks. The signer's eye gaze would follow the motion of the car.

- Before each of these classifier predicates, the signer would perform an ASL noun phrase which indicates which object is being referred to: HOUSE, CAT, or CAR. During this noun phrase, the signer's eye gaze would aim at the audience.

There are also some important elements of facial expression, head tilt, and other non-manual signals which convey meaning during a classifier predicate but are omitted from the current discussion.

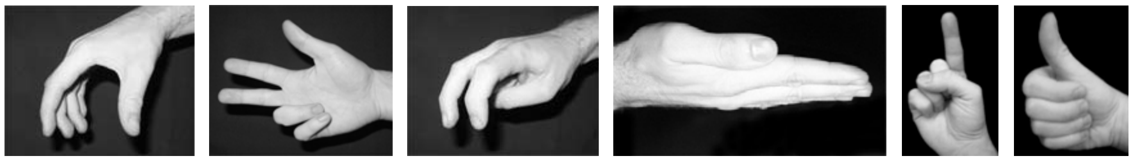


Figure 1: Classifier Handshapes: Spread C (bulky objects), Number 3 (motorized vehicles), Hooked V (animals or seated humans), Flat B (flat objects or surfaces), Number 1 (upright people), Thumb-Extended A (stationary objects)

Figure 2 contains a timing diagram of the performance of these three classifier predicates. The diagram records where the signer's eye gaze is aimed (in the "Gaze" row of the diagram), what the dominant hand is doing (in the "Right" row), and what the non-dominant hand is doing (in the "Left" row). The abbreviations "Loc#1," "Loc#2," and "Loc#3" are used to refer to the 3D locations in space in front of the signer's torso that have been associated with "the house," "the car," and "the cat" respectively. The location for "the car" is between the locations of "the house" and "the cat."

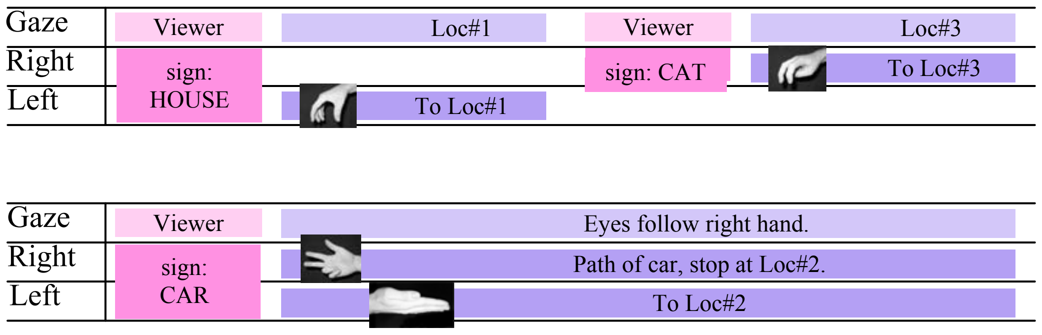


Figure 2: A Timing Diagram for a Classifier Predicate Performance.

The caption on Figure 1 describes the handshapes as being "classifier handshapes" – this reflects a difference between the terms "classifier" and "classifier predicate." A **Classifier** is the handshape chosen for the signer's hand when it is used to represent some entity in a 3D scene. The selection of what classifier to use is based upon: (1) the semantic properties of the object, (2) whether a CPML or SASS is being performed (see below), (3) the semantic details of the predicate being expressed, and (4) the layout of the 3D scene being discussed (Frishberg, 1975; Supalla, 1978; 1986; Emmorey, 2003; Brentari, 2005). A **Classifier Predicate** (CP) is a combination of a Classifier handshape and a 3D motion path for the hand (Liddell, 1977; DeMatteo, 1977; Mandel, 1977; Supalla, 1978).

There are several types of classifier predicates discussed in the linguistics literature. A **Classifier Predicate of Movement and Location** (CPML) is a classifier predicate which discusses the static position of objects in the depicting space or their movement through time (Supalla, 1978; Liddell, 2003a). These classifier predicates also indicate the orientation of objects, their arrangement in space, or ways in which they touch or physically interact with one another. (The use of the acronym CPML is unique to this document.) **Size and Shape Specifiers** (SASSs) are classifier predicates that describe the physical shape, appearance, structural configuration, or size of some object (Supalla, 1978). These classifier predicates are not the primary focus of this project; however, they have been considered during the design of our classifier predicate generation system (to allow for future growth).[5]

Signers can describe a complex scene involving multiple objects using a series of classifier predicates. For example, a signer who plans on using a CPML to show the movement or location of some object may first use a few additional CPMLs to set up the locations of other surrounding objects in the 3D scene being described.[6] This is the case in the "car parked between the cat and the house" example above. In more complex constructions, a signer may use their non-dominant hand to hold a location in a space associated with one object while they perform a classifier predicate with their other hand. In these "secondary object" constructions, the signer is able to more clearly show the spatial relationship between the two objects being described (Eccarius and Brentari, in press; Schick, 1987; Supalla, 1986). Signers can also produce complex classifier predicate performances in which they describe a 3D scene while alternating between CPMLs and other types of classifier predicates and/or switching between several different viewpoints/perspectives the scene.

[5] There are other types of classifier predicates during which the signer uses parts of his or her upper body to indicate parts of the lower body ("body part classifier predicates") or during which the signer may use his or her entire body to portray the body of a person in a scene being discussed (a "body classifier predicate"). Neither these nor other less-frequent subtypes of classifier predicates are the focus of this project.
[6] This type of classifier predicate construction will be within the scope of the prototype classifier predicate generator built for this dissertation project (see Chapter 8 for implementation details). The other complex classifier predicate constructions described in this section are beyond the scope of the prototype system.

The Surface-Form Representation: Articulators

An articulator is one of the small set of body parts which can move mostly independently to perform a classifier predicate. In this document, **Articulators** will actually be used to refer to the representation of each of these body parts within the system's surface-form representation. To represent an ASL performance, the following information about the Articulators must be recorded: the location/orientation/handshape of the dominant hand, the location/orientation/handshape of the non-dominant hand, the location at which the eye-gaze is aimed, the location at which the head-tilt is aimed, the direction of shoulder-tilt used to indicate roleshift, and several other non-manual signals (NMS) communicated via facial expressions or mouth shape. Many of these articulators can be thought of as moving through the space in front of the signer: the locations of the hands, the destination point of the eye gaze, the destination point of the head tilt, etc.

Difficulties in Using Standard MT for ASL

The way in which classifier predicates use the space around the signer to topologically indicate the locations and movements of objects in a 3D scene presents challenges not encountered in traditional written/spoken languages. Spoken languages do not visually present exact 3D locations, orientations, and motion paths of objects, and thus MT technology designed for these languages does not address how to generate such output. (Chapter 4 will discuss the unique challenges of generating classifier predicates.)

Not every ASL sentence contains a classifier predicate; in fact, the majority of ASL sentences do not. Further, traditional computational linguistic technology seems much better-suited to generating ASL LS sentences (at least compared to CP sentences). The ASL-unique challenges that arise during the generation of LS sentences include:

- Producing ASL pronouns (during which the signer points at a 3D location that has been associated with some object/concept under discussion)
- Modifying the motion path of ASL verbs (the beginning/ending of whose motion path indicates the verb's subject/object)
- Generating non-manual signals that indicate (or express agreement with) arguments in ASL, e.g. the marking of subject and object agreement on ASL verbs (by having head-tilt or eye-gaze directed at the location associated with these entities) (Neidle et al., 2000).[7]

While non-trivial, these spatial issues in the LS subsystem (which use Token Space) are less complex than those in the CP subsystem (which use Depicting Space). It is especially difficult to handle the requirement that objects be arranged in Depicting Space in a layout which is topologically analogous to a real-world 3D scene being discussed.

Another challenge for an English-to-ASL MT system is that it must produce a schedule for the movements of the various articulators of the signer's body during the animation to be performed by a virtual human character. Most MT technology for written languages is designed to produce text strings as output – Chapter 3 will discuss how string-like representations of language are not well-suited to encoding the parallel elements of an ASL performance or how these elements are coordinated over time.

[7] In recent research of Carol Neidle (personal communication, July 9, 2005), the presence of non-manual agreement over VP appears to correlate with VP focus, and it has thus been interpreted as focus marking instead of simply an expression of agreement.

Misconception:
It’s OK to Ignore Classifier Predicates

Even when MT researchers appreciate the distinct language status of ASL and try to build translation systems, they have chosen to focus exclusively on the LS subsystem of the language. Because the LS subsystem is easier to analyze and generate (since its less-complex use of space around the signer makes it somewhat closer in structure to known written languages), it has received a disproportionately large amount of attention from linguistic and MT researchers. (See Chapter 3 for a survey of several previous ASL MT systems.) Some ASL MT researchers have had success at producing ASL animations on this limited (non-spatial) portion of the language. Unfortunately, since none of these systems attempt to represent the spatial layout of objects in a 3D scene under discussion, they are unable to produce classifier predicates from an English text. Thus, no previous ASL MT system has designed or implemented a system for generating ASL classifier predicates.

Omitting classifier predicates from the output of an English-to-ASL MT system is not an appropriate or desirable simplification for several reasons. The first is that classifier predicates are actually quite common in fluent ASL signing. Studies of sign frequencies show that classifier predicates occur once per minute (and up to seventeen times per minute in some genres) (Morford & MacFarlane, 2003). Further, classifier predicates are the only way to convey some concepts contained in English sentences. For example, to express information about spatial layout, arrangement, shapes, outlines, alignment, or movement in ASL, a signer will use classifier predicates.

ASL User-Interfaces for the Deaf

Classifier predicates are particularly important when producing deaf-accessible user-interfaces for computer applications. Since ASL lacks a written form, any English on an interface would need to be translated into ASL and presented as a small animated character performing ASL on the screen. Clearly a computer application that involved spatial concepts would require classifier predicates in the ASL output, but more generally, these predicates are important in an interface because they enable the animation to refer to other elements on the screen. Since the ASL cannot be statically ‘written’ on elements of the interface, the dynamic animation performance will frequently need to refer to and describe elements of the surrounding screen.

When discussing a computer screen, a human ASL signer will typically draw an invisible version of the screen in the air with their hand and use classifier predicates to describe the layout of its components and explain how to interact with them. After the signer has “drawn” the screen in this fashion, he or she can refer to individual elements by pointing to their corresponding location in the signing space. Making reference to the onscreen interface is especially important when a computer application must communicate step-by-step instructions or help-file text. English-illiterate users of a computer application may also have limited computer experience; so, conveying this type of content can be especially important for them. For similar reasons, deaf children may benefit from computer user-interfaces that have been augmented with an on-screen ASL signing character.

Chapter 3: Previous Sign Language MT Research

This chapter will discuss previous research on the translation of English text into Signed English and into American Sign Language animations. In particular, the chapter will survey four English-to-ASL MT systems that come closest to the goal of automatic translation from English input text to fluent ASL animation output. The four systems will be compared on an issue-by-issue basis; each of these "issues" will be a major focus of improvement for the English-to-ASL MT design described later in this dissertation.[8]

English-to-Signed-English Systems

There have been some previous research systems designed to "translate" from English into Signed English or into a very English-like form of pseudo-ASL signing (Grieve-Smith, 2002; Bangham et al., 2000). As discussed in the previous chapter, deaf adults with low English literacy skills, who would be the target users of English-to-ASL machine translation software, generally do not find this form of English-like signing understandable. Since these systems gloss over the translation divergences between English and ASL, they are not discussed in great detail in this chapter.

[8] An earlier version of this survey is available as a technical report (Huenerfauth, 2003).

With little structural divergence between English text and Signed English animation, the architecture of most of these systems is fairly straightforward. Most involve a dictionary look-up process. For each word of the English input text string, the system will look up the correct sign in the Signed English dictionary, and create an animation which concatenates together all of the signs in the sentence into a complete animation. While the simple architectures of these systems do not inform the design of a linguistic translation component for an English-to-ASL system, these projects have developed some useful component technologies. Some have explored the design of sign animation databases (Furst et al., 2000) and software tools for linguistic informants to help build these databases (Wolfe et al., 1999). Some systems used motion-capture data-glove technology to collect 3D coordinates of a signing performance (Bangham et al., 2000; Ohki et al., 1994), and others generated sign animations from some form of symbolic encoding of each sign (Grieve-Smith, 2002; Lu et al., 1997). Other researchers have focused on the production of fingerspelling components (Davidson et al., 2000, 2001).

Unfortunately, some systems advertise themselves as “translation” systems and claim to produce “sign language” – thus misleading and disappointing software designers who are not aware of the differences between Signed English and ASL signing. As discussed in the previous chapter, technology that translates English text into English-like signing would not provide the same accessibility advantage to low-literacy deaf users that an actual English-to-ASL MT system would provide.

Four English-to-ASL MT Systems

This section will briefly introduce the four previous English-to-ASL MT systems that will be the focus of the remainder of the chapter. There have been other research projects that have considered the translation of written text into a signed language, such as: Polish Sign Language (Suszczanska et al., 2002), Japanese Sign Language (Ohki et. al, 1994; Lu et al., 1997; Tokuda & Okumura, 1998; Adachi et al., 2001), Chinese Sign Language (Xu & Gao, 2000), South African Sign Language (van Zijl & Barker, 2003), and Sign Language of the Netherlands (Verlinden et al., 2001). However, these projects have generally been short-lived, linguistically-simple, prototype systems; so, they will not be discussed in detail in this chapter. The four systems described below represent the most sophisticated and successful written-language-to-signed-language machine translation systems.

ViSiCAST

As part of the European Union's ViSiCAST project, researchers at the University of East Anglia implemented a system for translating from English text into British Sign Language[9] (Marshall & Sáfár, 2001; Sáfár & Marshall, 2001, 2002; Bangham et al., 2000). Their approach uses the CMU Link Parser to analyze an input English text, and then uses Prolog declarative clause grammar rules to convert this linkage output into a Discourse Representation Structure (DRS). During the generation half of the translation process, Head Driven Phrase Structure rules are used to produce a symbolic sign

[9] While British Sign Language is a unique sign language (with a different lexicon and structure than ASL – the two sign languages are not mutually intelligible), BSL does share enough fundamental linguistic properties with ASL that the ViSiCAST project has been included in this chapter.

language representation script. This script is in the system's proprietary "Signing Gesture Markup Language," a symbolic coding scheme for the movements required to perform a natural sign language (Kennaway, 2001).

ZARDOZ

The ZARDOZ system (Veale et al., 1998) was a proposed (and somewhat implemented) English-to-Sign-Languages translation system using a set of hand-coded schemata as an interlingua for a translation component. (The discussion herein will focus on the ambitious proposed architecture rather than the implemented portion.) While the implemented portion of the system focused on American Sign Language, the authors were developing their framework with British, Irish, and Japanese Sign Language also in mind. Some of the research foci of this system were the use of AI knowledge representation, metaphorical reasoning, and a blackboard system architecture; so, the translation design is very knowledge-based and AI-reasoning heavy.

During the analysis stage, English text would undergo sophisticated idiomatic concept decomposition before syntactic parsing in order to fill slots of particular concept/event/situation schemata. The advantage of the logical propositions and labeled slots provided by a schemata-architecture was that commonsense and other reasoning components in the system could later easily operate on the semantic information. Because of the amount of hand coding needed to produce new schemata, the system would only be feasible for limited domains. To compensate for these limitations, if a schema did not exist for a particular input text, the authors suggest that the system could instead perform word-for-sign transliteration (producing Signed English output).

ASL Workbench

The ASL Workbench was a proposed and partially implemented English-to-ASL machine-assisted translation system which incorporated modern research on ASL phonological models (Speers, 2001). The sophisticated phonological model used by this system is particularly robust and is based on the modern Movement-Hold model of ASL phonology (Liddell & Johnson, 1989). The design uses a lexical-functional grammar (LFG) for analysis of the English text into a functional structure, hand-crafted transfer rules for converting an English f-structure into an ASL one, and LFG rules to produce ASL output. The system uses a transfer-specific lexicon to map English words/phrases to analogous ASL signs/phrases.

When the system encounters difficulties in lexical choice or other translation tasks, it asks the user of the system for advice. The system also records a very simplistic discourse model from the English input (consisting of a flat list of discourse elements and their spatial locations – all of which are specified by the human operator). The ASL Workbench system was designed to be a tool to assist professional translators in converting from English to American Sign Language. In fact, the system cannot operate correctly without human intervention. It makes no attempt to resolve discourse referents in an English text; it requires the user to note when two phrases refer to the same entity.

TEAM

TEAM was an English-to-ASL translation system built at the University of Pennsylvania that employed Synchronous Tree Adjoining Grammar rules to build an ASL syntactic structure simultaneous to the parsing of an English input text (Zhao et al., 2000). The output of the linguistic portion of the system was a string-like ASL "gloss notation with embedded parameters" that encoded limited information about morphological variations, facial expressions, and sentence mood as features associated with the individual words in the string. This project took advantage of graphics research at the University of Pennsylvania by using one of the Center for Human Modeling and Simulation's (HMS) animated virtual humans as the signing avatar. In particular, this project explored how the HMS technology's ability to modify the "manner" of an animated character's movements (by modifying a small number of parameters in their animation software) could be used to modify the performance of some ASL signs to indicate adverbial information or perform morphological inflection.

ASL Representation: Phonological Detail

ASL is a language without a written notation system, making it a particularly difficult candidate for computational linguistic processing. It is theoretically possible for a machine translation system to analyze an English text and to immediately make decisions about the appearance of an ASL animation (such an approach would be analogous to a semantics-to-speech generation system for a spoken language). However, all practical approaches to English-to-ASL MT (and all four of the MT systems listed above) instead attempt to first construct a symbolic representation script of the ASL output to be performed. Without a standard ASL notation, the four systems have invented their own proprietary ASL script formats.[10]

[10] In fact, since the ASL Workbench never actually implemented the animation portion of the system, this symbolic ASL script is the final output of this system.

Any symbolic representation of an ASL performance will omit some amount of detail, and choosing what details are acceptable to omit when developing an artificial encoding scheme for a natural language is a challenging and error-prone task. Systems whose internal representation of ASL is insufficiently expressive cannot produce a fluent signing performance – since it may under-specify important phonological details. For instance, several of the ASL representations of the previous four MT systems have only attempted to encode a small amount of the non-manual signals of the ASL performance (the facial expression, eye gaze, head tilt, shoulder tilt, and other elements of performance aside from the movements of the hands).

The TEAM system's "glosses with embedded parameters" representation limited the quality of this system's ASL non-manual signals. The gloss representation is a string which contains the name of specific ASL lexical items in sequence. Some facial expressions can be represented by inserting "begin" and "end" markers into the string for a particular facial expression (such as "begin-eye-brow-raise" / "end-eye-brow-raise"). This representation fails to model how the degree of many forms of non-manual signals can change over time during an ASL performance.

TEAM's representation also limited its ability to perform phonological smoothing between ASL signs. The notation can attach features to particular glosses in the string to indicate special motion parameters for that particular ASL sign; however, the system stores the animation details for each ASL sign as separate 3D movement specifications (which are not accessible until very late in the generation process). The gloss notation does not separately record how the handshape, location, and orientation of each hand change over time – it just lists the discrete sequence of ASL signs in the sentence. Since many ASL phonological and morphological processes need to access this detailed form of ASL phonological information (Liddell & Johnson, 1989), this representation prevents the TEAM system from ever implementing these processes.

ZARDOZ's ASL representation is also insufficient in its treatment of non-manual signals. This system also represents the ASL sentence as a string of glosses for individual ASL signs to be performed. To add non-manual signals, the system inserts tokens into the stream to indicate a particular NMS (such as "eyebrows-downward"). A corresponding token is inserted later in the stream called "resume-previous-face." This token-insertion approach cannot represent the full spectrum of overlapped, simultaneous, and interacting non-manual signals of ASL. This token-insertion approach is similar to the "begin-specific-nms" / "end-specific-nms" approach used by the TEAM system, but it is even less powerful. The use of a "resume" token as opposed to a specific ending token for each type of NMS means that the ZARDOZ system assumes that all onsets and completions of NMS features will have a nested parenthesis structure. While this appears to generally be the case for ASL, it may not be for more complex ASL constructions or for all of the signed languages under consideration by the ZARDOZ project.

The ViSiCAST system uses a more detailed ASL animation script which includes phonological information encoded in the Signing Gesture Markup Language (Kennaway, 2001). This sign-language animation notation records greater detail about the performance of each sign; it separately records the details of each articulator of the ASL performance for each sign. Each lexicon entry in the system stores a phonological specification that can be modified during the generation process. This allows the ViSiCAST approach more flexibility than TEAM or ZARDOZ in morphologically modifying individual signs or linguistically blending the movements of adjacent signs. The system can consult and modify the specific performance details of signs – unlike TEAM or ZARDOZ, this system is not forced to treat each sign as an atomic unit in a sequence. (However, the initial implementation of this system generally does treat the signing performance specification as a string-like script of individual lexical items.)

The representation used by the Workbench system is based on the Movement-Hold ASL phonological model, and thus, the system does separately encode information about the location, orientation, movement, and handshape of each of the signer's hands during the ASL performance. This detail would enable this system to perform linguistically-accurate morphological and phonological modifications to the ASL signal. Unfortunately, the otherwise rich notation is very weak in its ability to model the NMS channel of output. While the specification for the hands is very detailed, the notation has only a single feature slot per time-slice for the NMS channel. This representation cannot model overlapping forms of simultaneous NMS or an NMS which changes in intensity during the performance.

Takeaway: We Should Encode Individual Articulator Details

We have seen how some ASL representations used by previous ASL MT systems leave out important information about some of the articulators (like the non-manual signals) or fail to encode sufficient detail about other articulators (like the separate values of location, orientation, and handshape for each of the hands). These representational failings have prevented these systems from developing linguistically-accurate morphological or phonological transformations which should operate on the ASL representation prior to animation performance.

In Chapters 6 and 7, we will discuss the ASL representation used in this English-to-ASL MT system. This representation will allow for the separate encoding of location, orientation, and handshape features for the signer's hands (and for other linguistic objects modeled by the system), and it will allow for the encoding of NMS features that can change in intensity and overlap during a performance (the prototype classifier predicate generator implemented for this dissertation includes two forms of NMS: the use of eye-gaze and eye-brow-raising during the performance of classifier predicates).

ASL Representation: Timing and Coordination

In addition to under-specifying the behavior of some of the articulators of the signing performance, the ASL representations used by the four previous MT systems have over-synchronized the timing relationships in the ASL performance. Fundamentally, all four of these systems have treated the ASL performance as a string of individual glosses (where a "gloss" is an English word that represents a particular ASL sign). Within this linear one-dimensional representation, these systems have tried to encode some of the other parallel/overlapping elements of the ASL performance.

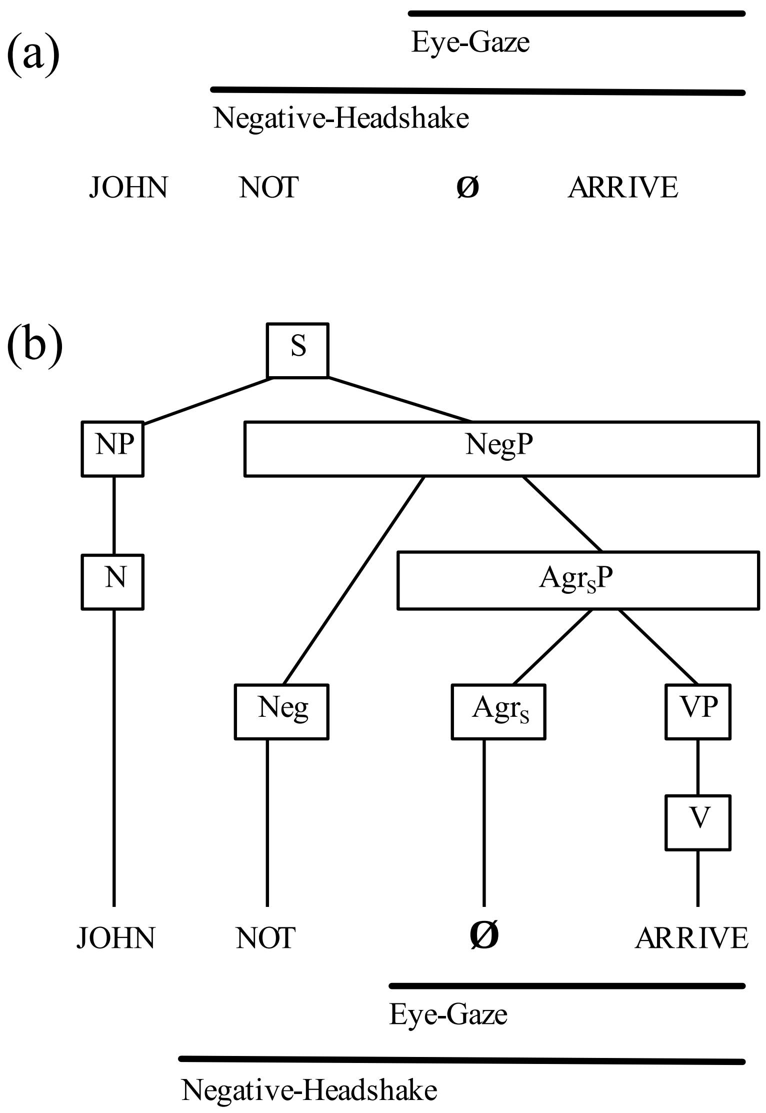
The TEAM and ZARDOZ systems are clearly string-based generation approaches; in both systems, the ASL sentence is represented as a sequence of ASL signs. Our earlier discussion of both systems has already revealed problems with their simplistic encoding of NMS (using beginning/ending tokens in the string). While both the ViSiCAST and Workbench systems did explicitly encode more intra-sign detail, the hierarchical (intersign) structure of the sentence was still represented as a syntax tree that assembled a sequence of glosses. Phenomena extending across multiple signs (like some NMS) were encoded using feature propagation in the tree (discussed later in this chapter).

Using strings to represent the supra-sign structure of ASL sentences allowed these systems to reuse MT technologies originally designed for written languages. However, the discussion below will show that strings (built using trees with feature propagation) fail to correctly capture the coordination timing relationships in the ASL signal.

How Linguists Represent ASL Performance

Linguists studying spoken languages often use string notations, but ASL linguists generally prefer a notation that records the multichannel nature of the signal. For example, let's consider how a particular ASL sentence has been glossed by syntactic researchers (Neidle et al., 2000). Figure 3(a) is an example of an ASL sentence written in a "decorated string" notation. During this sentence, signers produce three signs with their hands: JOHN, NOT, and ARRIVE. They shake their head (and furrow their brow) in a negative manner during the portion of the sentence under the "Negative-Headshake" bar. During the part of the sentence under the "Eye-Gaze" bar, signers' eye gaze aims toward a location in space that has previously been associated with JOHN.[11]

[11] Typically, signers use head-tilt to express subject agreement for ASL verbs, but eye-gaze can be used for intransitive verbs (Neidle et al., 2000). This expression of agreement is optional when the subject is expressed, and it may an indicator of VP focus (Carol Neidle, personal communication, July 9, 2005).



(c)

[[[JOHN]] [[NOT] [[**Ø**] [[[ARRIVE]]]]]]

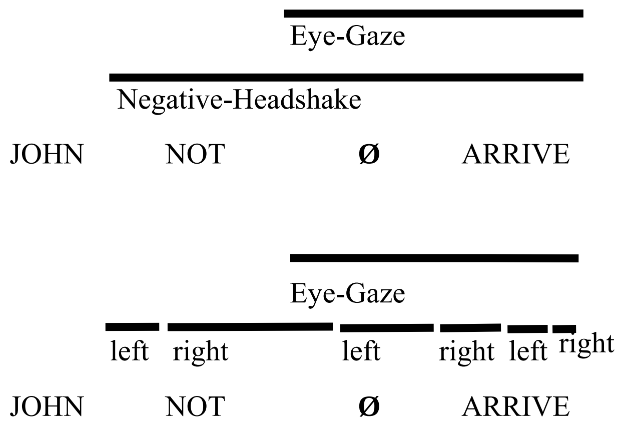
Figure 3: Example of ASL Linguistic Representations

The glosses (words) in the figure are not a writing system – ASL has no written form. The glosses are used by linguists to record the activity of an ASL signer's hands; each gloss is the closest English translation of a particular ASL sign (the set of glosses used to identify ASL signs is somewhat conventionalized). The dark bars above the glosses represent information not conveyed by the hands – non-manual signals (NMS). This notation has been called a "decorated string" because these bars can be thought of as "decorating" the string of glosses.

The notation in this example uses a "null symbol" (**Ø**) to act as a placeholder for a lack of lexical material (in this case, a manual sign) in a syntactic node. These "null symbols" are not generally included in the gloss of an ASL sentence, but it was included in this example by Neidle et al. (2000) to emphasize the syntactic node higher in the tree. (We have chosen to show an example containing a **Ø** symbol because we will be using a similar "null" notation later in this chapter for a different purpose.)

While the **Ø** does not make any claim about the timing of the performance, in the sentence shown in Figure 3, the Eye-Gaze does generally start a moment before the beginning of the manually performed sign ARRIVE. In the example, there is also non-coordination between the headshake and the string of words: A headshake consists of a series of individual left and right head movements – the notation does not encode how each movement should be coordinated with each sign in the sentence performance.

Consider the two notations shown in Figure 4; the second diagram gives an informal decomposition of the individual movements of a Negative-Headshake to illustrate how they need not align in a precise way with the boundaries between individual signs.



The original linguistic annotation of the sentence using a single horizontal bar to represent the Negative-Headshake.

In this version of the diagram, the Negative-Headshake is shown as a set of individual left and right movements of the head.

Figure 4: Decomposition of Negative-Headshake into Left and Right Movements

As we can see in Figure 4, the entire headshaking event is composed of a series of left-to-right and right-to-left movements of the head. Minor variations in the timing of these individual movements (relative to the manual parts of the performance) would not produce different meanings – they would reflect linguistic variation.[12] Because the "decorated string" doesn't specify how these individual movements are linked to the string, several performance variations are represented by the notation – the head movements are not coordinated to the individual sign boundaries.[13]

Since we would like to develop a representation of the internal structure of an ASL signal, it is useful to consider ASL syntactic tree representations used in the linguistic literature. Consider the syntax notation for this sentence in Figure 3(b). (This figure

[12] While it may not be clear in Figure 4, it is actually the first "right" movement of the head that is coordinated with the start of the ASL sign "NOT." In fact, the first "left" head movement may actually be an anticipatory movement needed to get the signer's head in the proper position to begin the "Negative Headshake" performance. These timing subtleties are not discussed further in this dissertation.
[13] While the left and right movements of the head are not coordinated with other parts of the performance in this example, there could be coordination between periodic non-manual and manual movements in other sentences. For example, the movements of the head could be coordinated with the movement path of the hands during certain types of aspectual verb inflection.

contains a simplified version of a syntactic analysis that follows the style of analyses given in (Neidle et. al, 2000); the linguistic details of this particular analysis are not important for the way the example is used in this chapter.) The tree explains the arrangement of the manual signs, but it doesn't indicate how the NMS bars are linked to them. Since trees are a graphical way to represent a nested structure for a text string, consider how the tree can be represented as a (one-dimensional) bracketing structure in Figure 3(c). The NMS bars would extend beyond the constituents in the brackets – it's not clear how the bars fit into the notation.

Strings of Glosses with Feature Propagation

While decorated strings and syntax trees have been useful tools for ASL linguists, an ASL animation system may need a more precise representation of the timing coordination relationships in the ASL sentence. Since the system must produce "phonetic" output of an actual ASL animation, it is important to distinguish which temporal coordination relationships are important to maintain in the final animation (and which are not).

The four MT systems in this chapter have encoded the ASL signal using strings of glosses, and they have used a traditional NLP grammar modification called "feature propagation." They associate values with nodes in the syntax tree, and these "features" spread their value from parents down to children to "propagate" information through the tree. In Figure 5, one feature [+shake] indicates headshake is occurring and another [+gaze] indicates eye-gaze. These features are passed down to the individual signs at the leaves of the tree. Note how the Agr_SP node passes the [+shake] from its parent to its children and adds a [+gaze] to all its descendents.

There are problems with this "string of glosses with feature propagation" approach. Specifically, it splits the representation of the NMS bars into individual events for each sign. The ASL performance is treated like a string of beads on a chain – the boundary between each sign is treated as a global synchronization point for all of the ASL articulators. While the notation encodes the NMS, it doesn't accurately represent the coordination relationships in the signal. We discussed above how the negative headshake movements are not coordinated with the boundaries of individual signs, yet this notation implies that there are individual headshake events coordinated with each sign. In an ASL generation system, a final output module would need to process this tree representation and produce an animation of a signer. While the system could merge these individual headshake features together into a single multi-sign event, the representation doesn't indicate when it's allowed to merge features across signs (and when it cannot).

Takeaway: Strings are Poor Encodings of ASL Coordination

The ASL representations used by these four systems do not capture the temporal relationships in the signal at a proper level of detail and flexibility. Their string-based encodings of ASL sentences artificially linearize the signal before it is necessary – they introduce a global synchronization point at the boundary between every pair of signs. This makes it difficult to accurately represent which temporal relationships between parallel ASL phenomena must be coordinated (are linguistically required to be) and which can be left non-coordinated.

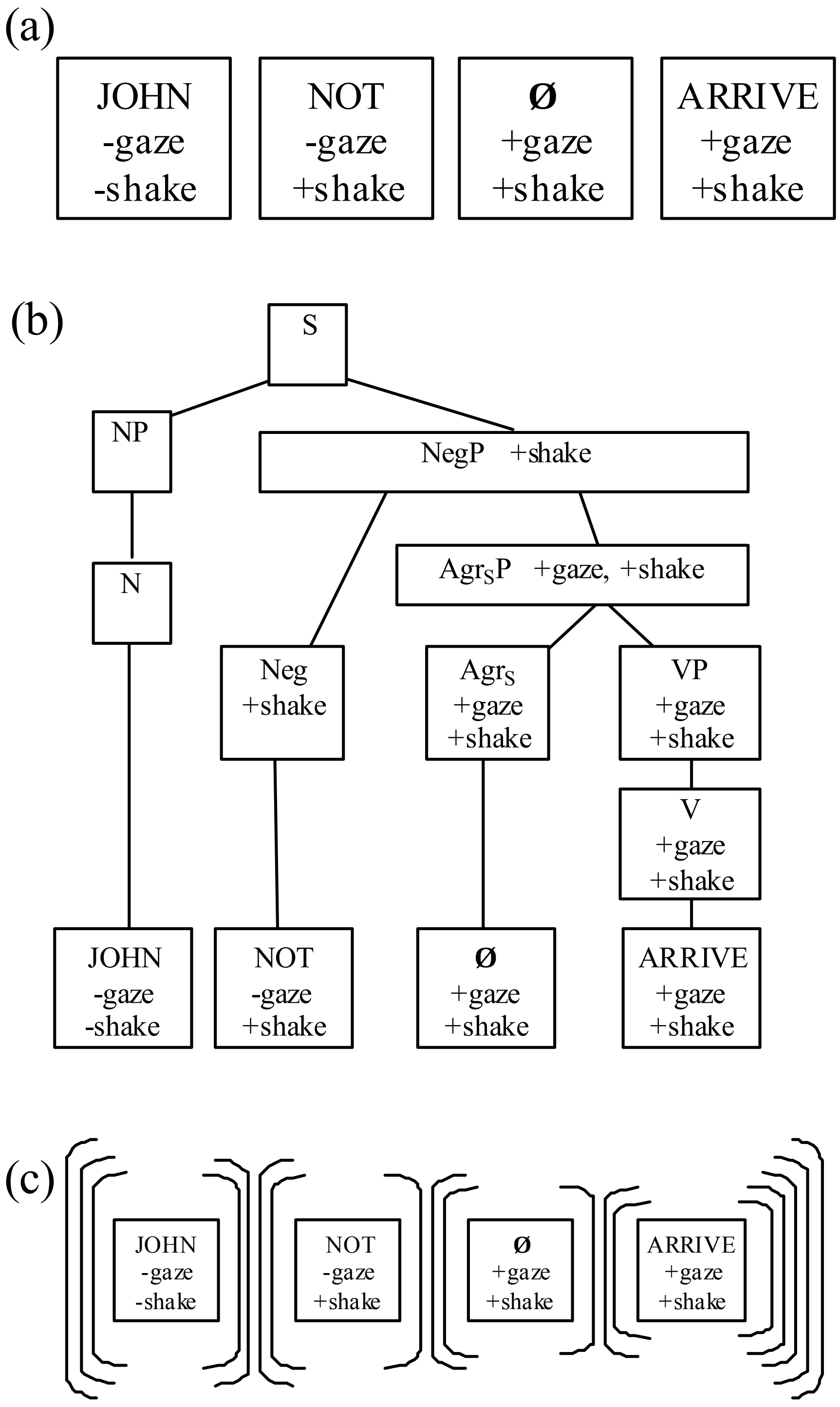


Figure 5: String of Glosses with Feature Propagation

By over-coordinating the signal, these systems produce an overly constrained script of the movements required of the animated human character. It is already a challenging task for an animated character to quickly perform the sometimes-competing movements that a script of an ASL performance might require. Introducing unnecessary constraints into this script could make the character's task too difficult to be completed successfully (or force the character to sacrifice naturalness of movement to meet the requirements).

Chapter 6 will discuss the development of a new multichannel ASL representation (used in the English-to-ASL MT design of this project) that can encode both coordination and non-coordination relationships between parallel portions of the ASL performance.

MT Architectures and their Tradeoffs

There is a spectrum of architectural designs along which most machine translation systems can be classified, and loosely they are grouped into three basic designs: direct, transfer, or interlingua (Dorr et al., 1998). Direct systems process the words of the source language string without performing syntactic analysis on the text. Transfer systems do analyze the input text to some syntactic or semantic level, and then a special set of "transfer" rules are employed to read the information of the source language structure and produce a corresponding syntactic or semantic structure in the target language. Afterwards, a generator converts this linguistic structure into a target-language surface form. Interlingua systems analyze the input text one step further: the source is analyzed and semantically processed to produce a language-independent semantic representation called an interlingua, and then a generation component produces the target-language surface form from this starting point. These design choices are often pictured as a pyramid, as in Figure 6, adapted from a figure in (Dorr et al., 1998).

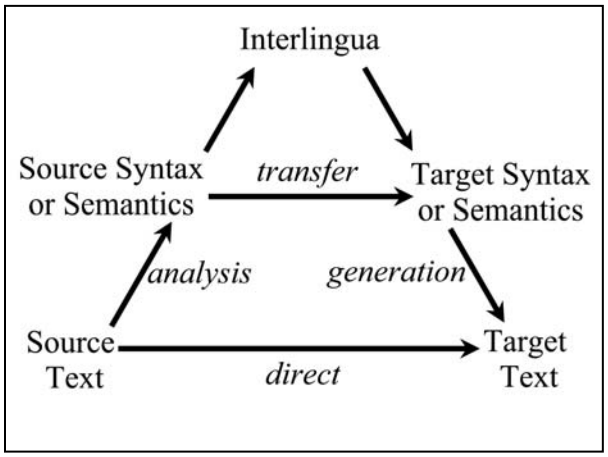


Figure 6: The MT Pyramid of Architecture Designs

The systems mentioned at the beginning of this chapter (which translated English into Signed English or into English-like pseudo-ASL signing) used a "direct" approach. The processing architecture of these systems organized the translation task as a word-to-sign replacement process. For example, the TESSA system (Bangham et al., 2000) takes an English input text, looks up each word of the English string in its English-to-Sign dictionary, concatenates the animation for each of these signs together, and blends them into a full animation. Since a "direct" system does not represent any of the underlying linguistic structures of English or ASL, the translation divergences between the two are difficult to handle. TESSA used a small set of standard sign language sentence templates to attempt to rearrange the text into an form which was slightly less English-like, but the use of templates in this manner is not scalable. For this reason, this system could never translate from English into a natural sign language, and instead merely encodes an English string visually as Signed English.

Another MT system using a lexical-based translation architecture was TEAM. This project used a synchronous lexicalized TAG grammar for English and for ASL, where a bilingual lexicon stored English word / ASL sign pairs. Each of the members of the pair stored a piece of TAG structure that represented the syntactic environment surrounding the lexical item in which that particular bilingual word/sign pairing would be valid. As the English input text was analyzed with a TAG parser, the English lexical entries were looked up, and the ASL signs to which they corresponded, identified. Simultaneous to the TAG analysis, the TAG generator for ASL worked to assemble the ASL TAG trees into a complete sentence. Because of the increased linguistic information, this architecture could handle many syntactic divergences between English and ASL.

While the TEAM approach may sound like a direct architecture since there seems to be a word-to-sign mapping, it is actually a syntactic transfer approach. The English input string needs to be analyzed with the TAG parser during the translation process, and the syntactic information revealed helps direct the bilingual lexicon look-up process. The "transfer rules" in this system would be each of the paired entries in the bilingual lexicon; by identifying and applying this matching process, TEAM converts a syntactic analysis of the English sentence into a syntactic structure for ASL.

Another English-to-ASL machine translation tool using a "transfer" architecture was the ASL Workbench system. This system performed deeper linguistic analysis than TEAM; instead of stopping at a basic syntactic constituent structure, this system analyzed the input English text up to a Lexical Functional Grammar f-structure. This representation abstracts away some of the syntactic specifics of the input text and replaces them with linguistic features of the text, such as “passive voice,” or functional labels on constituents, like “subject” or “direct object.” This Workbench system included a set of specially written rules for translating an f-structure representation of English into one for ASL.

While this system’s deeper level of analysis may help it handle some more subtle forms of divergence than a system with more shallow analysis of the English source, the biggest advantage of analysis up to f-structure is that it simplifies writing transfer rules. Instead of conditioning on constituent structures of a syntactic tree, the rules reason about abstract features or functional role labels in the English f-structure. Another advantage of deeper transfer is that generation for the ASL side begins with a more abstract representation. While this may sound like more work for the ASL generator, it actually gives the system more flexibility in its choice of structure.

The ViSiCAST system also used a transfer-based architecture; in this case, the cross-over from the English analysis to the sign language generation occurs at a semantic level of representation. The English portion of the system converts a text into a set of variables and semantic predicates in a Discourse Representation Structure. After a conversion process to a hierarchical semantic format, a generation system begins the sign language production process. The types of divergences that can be handled by such a system are more sophisticated than those that the syntactic-transfer TEAM or Workbench systems can accommodate; the use of semantic predicates helps to break apart the individual elements of meaning in the English source text and helps avoid an English syntactic influence on the sign language output being generated. However, these additional levels of representation and syntactic-semantic linkages significantly increase the development time of this system compared to those approaches that do not go as high up the machine translation pyramid.

The ZARDOZ system falls into the final category of English-to-ASL translation systems – those which use an interlingua representation. The authors of this system identified many types of semantic divergences between English and ASL which could only be successfully handled by a system that would be able to employ some form of spatial or commonsense reasoning. (Most of these issues surround the generation of classifier predicates.) So, in their proposal, the input English text would be analyzed syntactically and semantically, and this information would be used to select a particular event schema. These schemata would record all the common types of events and actions occurring in the world (and their participants); so, the time needed to develop this set of schemata restricts this system to a small domain. If a schema were filled using the information from the English string and any world knowledge, then a generation component could be used to express the information stored in the schema in ASL animation output. Because there was a common representation structure used for both English and ASL, and because these schemata were generally language-independent, they can be considered an interlingua.

The final section of this chapter will call into question whether ZARDOZ's architecture could actually be used to create a successful classifier predicate generator – a different design that relies on less AI-reasoning will be described in Chapter 4.

Takeaway: Some Architectures are Better for SE, LS, or CP

We've seen that the different non-statistical MT architectures are each capable of producing different types of signing output: direct approaches work well for Signed English (SE), transfer approaches seem to work well for most LS signing, and deeper (interlingua based) approaches may be needed for CP signing. The increased divergence-handling power of some architectures comes with a cost – increased development time. As you go higher up the MT Pyramid, additional linguistic resources need to be built. There is a divergence-handling-power vs. development-effort trade-off.

Since an English-to-ASL MT system may at times need to produce all three different forms of signing (SE, LS, and CP), then it might seem like we are forced to use a complex interlingua-based MT design to guarantee that we can generate all three. Chapter 5 will discuss how our MT system plans to get around this trade-off by using a multi-path architecture (with a different generation pathway for ASL CP, for ASL LS, and for SE animation output). A direct, a transfer, and an interlingua-based MT processing pathway will be combined into a single hybrid design. Of course, the ASL representation used in the system will need to be able to represent all three types of signing output and to interact with the generation pathway that is used for each.

Generation of Classifier Predicates

Chapter 2 discussed how the complexity and variability of classifier predicates has led most ASL machine translation researchers to omit classifier predicates from their systems. While no ASL MT system has been implemented capable of producing classifier predicates, two of the ASL MT systems (ASL Workbench and ZARDOZ) have partially addressed how classifier predicates could be included in their design.

While the designers of the ASL Workbench system did not explain how to produce a classifier predicate from an English text, they did discuss how classifier predicates could be incorporated into its lexicon. Classifier predicates would be stored as highly underspecified lexical entries (morphological "roots") whose movement path would be determined by the generation grammar. While the documentation for the ASL Workbench only included a brief discussion about how it might handle classifier predicates in this way, the idea of representing classifier predicates as underspecified templates is also used in the design of the English-to-ASL MT system in this thesis.

ZARDOZ was only a partially-implemented proposed design, but the proposal did include some ambitious ways in which classifier predicates could be generated by the system's interlingua architecture. In this design, particular styles of classifier predicate expression could be represented by unique interlingua "frames" that could be selected and filled by the analysis/understanding portion of the translation architecture. The proposal discussed in general terms how spatial and commonsense reasoning approaches could be used to fill in the animation details needed to produce a fluent classifier predicate handshape and movement. Considering the rather unconstrained structure of the interlingua "frames" proposed by the ZARDOZ designers, it would be prohibitively time-consuming to develop an AI-reasoning-based system to produce classifier predicates. Further, the design doesn't specify limits on the amount of knowledge-dependence or the amount reasoning the "frames" may require during generation.

Takeaway: We Need a Way to Generate CPs

Chapter 2 has already discussed why an English-to-ASL system should be able to produce classifier predicates, yet they were not included in the output of any of these four ASL MT systems. Chapter 4 will argue that in order to generate classifier predicates, it is necessary to visualize a 3D model of the layout of the scene being discussed. None of the four ASL MT systems in this chapter included scene-visualization technology in their design.

This chapter has also criticized the proposal of the ZARDOZ system that classifier predicates should be generated using complex AI-reasoning that relies on a large knowledge-base of information. Chapter 4 will describe a classifier predicate generation approach that instead uses a planning-operator formalism to restrict the expressivity and knowledge-dependence of the templates representing classifier predicates. A simpler, planning process will be used to fill these templates and produce an animation output.

One reason why more complex reasoning was needed in the ZARDOZ system is that this system did not make a distinction between the visualization of a 3D scene from an English text and the production of a classifier predicate animation[14]. The classifier predicate generator described in Chapter 4 will isolate the scene-visualization process from the classifier predicate generation (the second process can be implemented within the relatively simpler processing mechanism of a planning algorithm).

[14] The system didn't possess a 3D model, but it could have potentially encoded the layout of the scene in the logical predicates in its knowledge base representation. The system also didn't produce any actual classifier predicate animations, but the designers did discuss how they might have been produced in general terms. So, this analysis of the classifier predicate capabilities of ZARDOZ is somewhat hypothetical.

Chapter 4:
Generating ASL Classifier Predicates

A unique aspect of ASL performance which is not generally shared by spoken languages is the complex, systematic, and conventional way in which the space around the signer is used to convey meaning. While an English speaker may use gestures during a conversation, their meaning is often vague, underspecified, or redundant to the spoken channel. In ASL, the way in which portions of the space around the signer are assigned specific meanings is an essential feature of the language signal. Chapter 2 discussed several motivations for designing an English-to-ASL machine translation system that is capable of generating animations of ASL classifier predicates. This chapter will discuss the complex ways in which ASL uses the space around the signer to convey information, and it will describe a new architectural design for generating classifier predicates.

Specifically, this chapter will focus on how an English-to-ASL MT system could successfully generate ASL classifier predicates of movement and location (Supalla, 1982; Liddell, 2003a). This chapter will assume that English input sentences that should be translated into ASL classifier predicates have been identified. Chapter 5 will discuss how this is possible within a "multi-path" machine translation architecture.

Classifier Predicates of Movement and Location

Classifier predicates allow signers to use their hands to position, move, trace, or reorient an imaginary object in the space in front of them to indicate the location, movement, shape, contour, physical dimension, or some other property of a corresponding real world entity under discussion. Classifier Predicates of Movement and Location (CPMLs) convey the movement and location of these objects (generally without indicating their specific size or shape). A CPML consists of a semantically meaningful handshape and a 3D hand movement path. The handshape is chosen from a closed set based on characteristics of the entity described (whether it is a vehicle, human, animal, etc.) and what aspect of the entity the signer is describing (surface, position, motion, etc).

For example, the English sentence "the car drove down the bumpy road past the cat" could be expressed in ASL using two classifier predicates. First, a signer would move a hand in a "Hooked V" handshape (see Figure 1 on page 21) forward and slightly downward to a point in space in front of his or her torso where an imaginary miniature cat could be envisioned. Next, a hand in a "Number 3" handshape would trace a path in space past the 'cat' in an up-and-down fashion as if it were a car bouncing along a bumpy road.

The ability of classifier predicates to topologically represent a three-dimensional scene makes them particularly difficult to generate using traditional computational linguistic methods and models. To produce this pair of classifier predicates, this chapter will argue that there must be a spatial model of how the scene is arranged including the locations of the cat, the road, and the car.

Competing Linguistic Theories of CPMLs

Classifier predicates challenge traditional definitions of what constitutes linguistic expression; they oftentimes incorporate spatial metaphor and scene-visualization to such a degree that there is debate as to whether they are paralinguistic spatial gestures, non-spatial polymorphemic constructions, or compositional yet spatially-parameterized expressions (Liddell, 2003b). No matter their true linguistic nature, an English-to-ASL MT system must somehow generate these expressions.

While MT designs can be successful without mirroring linguistic models of human language production, it is worthwhile to consider linguistic models that account well for the ASL classifier predicate data but minimize the computational or representational overhead required to implement them. The next section of this chapter will examine some approaches to generating ASL classifier predicates that directly correspond to some of the common linguistic theories of their structure. We will examine each of these approaches in turn, and we will discover challenges with adopting each of them as a computational approach for an MT system.

We will first consider a classifier predicate MT approach requiring little linguistic processing or novel ASL representations, namely a fully lexicalized approach. As engineering limitations are identified or additional linguistic analyses are considered, the design will be modified, and progressively more sophisticated representations and processing architectures will emerge throughout this chapter.

Can We Generate CPs without Visualizing the Scene?

The design of our English-to-ASL MT system will incorporate a component that will visualize the 3D arrangement of a set of objects being discussed by an English text (and then use this information to help generate classifier predicates describing the movement and location of these objects). Visualizing the 3D arrangement of a scene based on an English text description is clearly a non-trivial task, and so we would like to be certain that such an approach is really necessary in our MT system. The two generation approaches discussed in this section do not require such a visualization of the scene; we will examine these two approaches and uncover problems with each.

Fully Lexicalizing the Movement Paths

The task of selecting the appropriate handshape for a classifier predicate, while non-trivial, seems approachable with a lexicalized design. For example, by storing semantic features (e.g. +human, +vehicle, +animal, +flat-surface) in the English lexicon, possible handshapes can be identified for entities referred to by particular English nouns. Associating other features (e.g. +motion-path, +stationary-location, +relative-locations, +shape-contour) with particular verbs or prepositions in the English lexicon could help identify what kind of information the predicate must express – further narrowing the set of possible classifier handshapes. So, careful design of an English-to-ASL translation lexicon could allow us to successfully select the proper handshape for a CPML.

To produce the 3D hand movement path of a CPML using a fully-lexicalized approach, we would need to store sets of 3D coordinates in the English lexicon for each word or phrase (piece of lexicalized syntactic structure) that may be translated as a classifier predicate. Unfortunately, the highly productive and scene-specific nature of these signs makes them potentially infinite in number. For example, while it may seem possible to simply store a 3D path with the English phrase "driving up a hill," factors like the curve of the road, steepness of hill, how far up to drive, etc. would affect the final output. So, a naïve fully-lexicalized treatment of CPML movement would not be scalable. The variation in classifier predicate performances based upon the specifics of the scene being discussed would make this approach combinatorially impractical.

Morphologically Composing the Movement Paths

Since the system may need to produce innumerable possible classifier predicates, we can't merely treat the movement path as an unanalyzable whole. A more practical design would compose a 3D path based on some finite set of features or semantic elements from the English source text. This approach would need a library of basic animation components that could be combined to produce a single classifier predicate movement. This animation library would contain common positions in space, relative orientations of objects in space (for concepts like above, below, across from), common motion paths, or common contours for such paths. Finally, these components would be associated with corresponding semantic features or lexical items of English so that the appropriate animation components can be selected and combined at translation time to produce a full 3D path for the signer's hand.

This design is analogous to the polymorphemic model of classifier predicate generation (Supalla 1978, 1982, 1986). This model describes ASL classifier predicates as categorical, and it characterizes their generation as a process of combining sets of spatially semantic morphemes. As more and more of these morphemes are combined, the signer eventually "narrows in on" a specific movement path to use for the CPML. The difficulty with this approach is that every piece of spatial semantic information we might express with a classifier predicate must be encoded as a morpheme. Classifier predicates can convey such a wide variety of spatial information – especially when used in combination to describe spatial relationships or comparisons between objects in a scene – that very many morphemes are required.

For example, Liddell's analysis (2003b) of the polymorphemic model indicates that in order to generate the variety of classifier predicates seen in ASL data, the model would need a tremendously large (and possibly infinite) number of morphemes. Using a polymorphemic analysis, Liddell (2003b) decomposes single classifier predicate (of one person walking up to another), and he finds over 28 morphemes, including some for: two entities facing each other, being on the same horizontal plane, being vertically oriented, being freely moving, being a particular distance apart, moving on a straight path, etc. Liddell considers classifier predicates as being continuous and somewhat gestural in nature (2003a), and this partially explains his rejection of the model. (If there are not a finite number of possible sizes, locations, and relative orientations for objects in the scene, then the number of required morphemes becomes infinite.)

Whether classifier predicates are continuous or categorical and whether this number of morphemes is infinite or finite, the number would likely be intractably large for an MT system to enumerate and process. We will see that the CPML generation design described at the end of this chapter will use a non-categorical approach for selecting its 3D hand locations and movements. This should not be taken as a linguistic claim about human ASL signers (who may indeed use the large numbers of morphemes required by the polymorphemic model) but rather as a tractable engineering solution to the highly productive nature of classifier predicates.

Another reason why a polymorphemic approach to classifier predicate generation would be difficult to implement in a computational system is that the complex spatial interactions and constraints of a 3D scene would be difficult to encode in a set of compositional rules. For example, consider the two classifier predicates in the "the car drove down the bumpy road past the cat" example. To produce these predicates, the signer must know how the scene is arranged including the locations of the cat, the road, and the car. A path for the car must be chosen with beginning/ending positions, and the hand must be articulated to indicate the contour of the path (e.g. bumpy, hilly, twisty). The proximity of the road to the cat, the plane of the ground, and the curve of the road must be selected. Other properties of the objects must be known: (1) cats generally sit on the ground and (2) cars generally travel along the ground on roads. The successful translation of the English sentence into these two classifier predicates involved a great deal of semantic understanding and spatial reasoning. It would be quite difficult to design a set of rules that capture all of these spatial relationships and interactions. For this reason, our English-to-ASL MT system will instead visualize the actual motion path of the car (and use this information to select a motion path for the signer's hand).

How Can We Visualize a 3D Scene?

ASL signers using classifier predicates handle the complexities mentioned above using their own spatial knowledge and reasoning and by visualizing the elements of the scene. An MT system can also benefit from a 3D representation of the scene from which it could calculate the movement paths of classifier predicates. While the approach mentioned above used compositional rules (and associated morphemes) to cover every possible combination of object positions and spatial implications as suggested by English texts, the CPML-generation approach of our system uses virtual reality 3D scene modeling software to simulate the movement and location of entities described by an English text (and to automatically manage their spatial interactions).

AnimNL: A Scene Visualization System

A software system for producing a changing 3D virtual reality representation of a scene from an English text has already been developed: the AnimNL system (Schuler, 2003; Bindiganavale et al., 2000; Badler et al., 2000). The system displays a 3D animation and accepts English input text containing instructions for the characters and objects in the scene to follow. It updates the virtual reality so that objects obey the English commands. AnimNL has been used in military training and equipment repair domains and can be extended by augmenting its library of Parameterized Action Representations (PARs), to cover additional domains of English input texts.

The system's ability to interact with language and plan future actions arises from the use of PARs, which can be thought of as animation/linguistic primitives for structuring the movements in a 3D scene. PARs are feature-value structures that have slots specifying: what agent is moving, the path/manner of this motion, whether it is translational/rotational motion, the terminating conditions on the motion, any speed or timing data, etc. A single locomotion event may contain several sub-movements or subevents, and for this reason, PARs may be defined in a hierarchical manner. A single "high-level" PAR may specify the details for the entire motion, but it may be defined in terms of several "low-level" PARs which specify the more primitive sub-movements/events.

The system stores a database of PAR templates that represent prototypical actions the agent can perform. These templates are missing particular details (some of their slots aren't filled in) about the position of the agent or other entities in the environment that would affect how the animation action should really be performed in particular situations. By parameterizing PARs on the 3D coordinates of the objects participating in the movement, the system can produce animations specific to particular scene configurations and reuse common animation code.

English lexicalized syntactic structures are associated with PARs so that the analysis of a text is used to select a PAR template and fill some of its slots. For example, there may be a PAR associated with the concept of "falling" vs. another for "jumping." While these templates must remain parameterized on the 3D location of the agent of the movement until it is known at run time, there are some properties (in this case, the direction of motion) that can be specified for each from the English semantics. During analysis of the English input text, semantic features of motion verbs are obtained from the VerbNet hierarchy (Kipper et al., 2004), and these features are also used to select and fill a particular motion template. Since VerbNet groups verbs that share common semantic/syntactic properties, AnimNL is able to link an entire set of semantically similar motion verbs to a single PAR template. Each of the verbs in the set may fill some of the slots of the motion template somewhat differently.

When a PAR template has been partially filled with information from the English text and 3D object locations, it is passed off to AnimNL's animation planner. In fact, PARs contain slots allowing them to be hierarchical planning operators: pre-conditions, effects, subplans, etc. The movements of all objects in the AnimNL system are governed by a planning process, which allows the objects in the scene to move realistically. Many spatial motions have conditions on the location, orientation, or motion state of an object and its environment before, during, and after the event. The PAR operators help the system work out the details of an animation from the limited specification of this motion provided by an English text. For example, it may determine starting and stopping locations for movement paths or select relative locations for objects in the 3D scene based on prepositions and adverbials in the English input text. The interaction and conditions of these planning operators simulate physical constraints, collision avoidance, human anatomical limitations, and other factors to produce an animation.

Using 3D Visualizations in an ASL MT System

AnimNL is not the only software system designed to produce a 3D scene from an English text description. For instance, the WordsEye project (Coyne and Sproat, 2001) has also studied how to create 3D models from texts. Thus, our MT system could use either of these scene-visualization programs (or others) to build a 3D model of the movement of a set of objects from an English text discussing locations and movements.

The scene visualization software would analyze an English input text, and based on this analysis, it would create a 3D representation of the location and motion of the entities in the text. Once an animated 3D model of the scene has been produced, it can be used in our English-to-ASL MT system in the following way: (1) the scene will be miniaturized (and possibly rotated), (2) the objects in the scene will be made invisible, and (3) the scene will be "overlaid" onto the space in front of the ASL signing character.

In this way, a miniature 3D virtual reality would be embedded within the original 3D space containing the standing animated virtual human. (See Figure 7.) In this example, the sentence "The car parked next to the house" would be analyzed by the scene visualization software to produce a 3D animation of a car parking next to a house. This 3D scene will be mapped onto the space in front of the signer's torso, and the objects in the scene will be converted into invisible placeholders (representing the location and perhaps the orientation of the object). The color or other precise visual details of the house and the car are not important for the system to get correct – this information does not affect the performance of classifier predicates of movement and location.

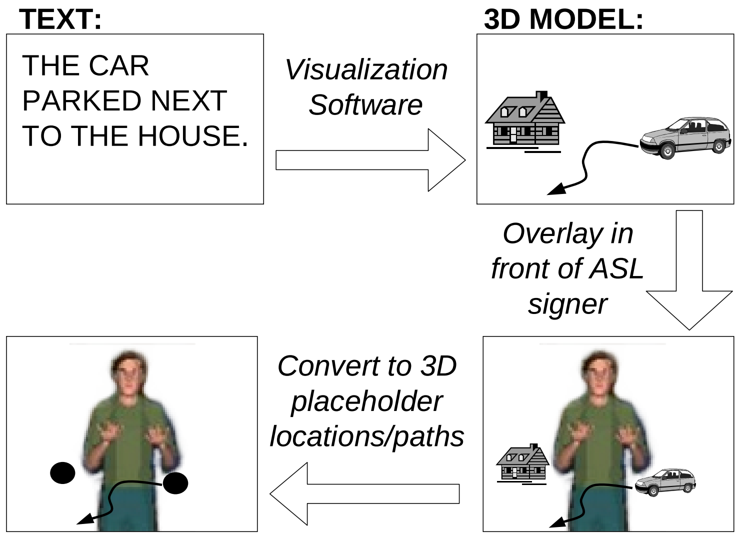

TEXT:
THE CAR PARKED NEXT TO THE HOUSE.
Visualization Software
3D MODEL:
Overlay in front of ASL signer
Convert to 3D placeholder locations/paths

Figure 7: Use of Scene Visualization in ASL MT

One challenge for scene visualization systems is that English sentences are often ambiguous about the details of the spatial arrangement of objects that they discuss. When we say "the car parked near the house," we don't necessarily know whether the car was facing the house when it parked or whether it pulled up along side it. The scene visualization system will often need to select one of these options when producing the 3D model, and it may not have sufficient information to make this decision. If there are errors in the way in which the scene visualizer arranges the objects in the 3D model, this error will be propagated down the processing pipeline and may become apparent in the classifier predicate performance that is produced. In this case, an error in the orientation of an object may not be apparent if certain classifier handshapes are used which do not indicate a front/back for the object being depicted.

Of course, even a human English-to-ASL interpreter who listens to an English sentence may make an error when setting up the objects in space around her during a classifier predicate performance. For instance, if an interpreter heard the sentence "Bob walked around the field," she might not be able to decide whether Bob wandered around inside of the field or whether he walked around the perimeter of the field. While a human interpreter can consider the context of the sentence and her world knowledge, she may still make errors (just like a piece of scene visualization software might) if the English sentence is truly ambiguous about the spatial details. In this case, the interpreter (like the scene visualizer) will have to guess how the scene was arranged. Since ASL classifier predicates are often less ambiguous about spatial arrangements, then this "guessed" information will become part of the message that is conveyed.

Obtaining Scene Visualization Data from Other Sources

In other applications, the 3D layout of the objects being discussed may be easier to calculate or already provided to the system. If the English sentence to be translated is discussing objects whose spatial locations are already known to the computer, then the MT system would not need scene-visualization software. For instance, in a user-interface for a computer system, an English sentence discussing the layout or movement of GUI screen elements could directly copy the 2D locations of the objects to build a model of how they are located and move in the scene (Huenerfauth, 2005c).

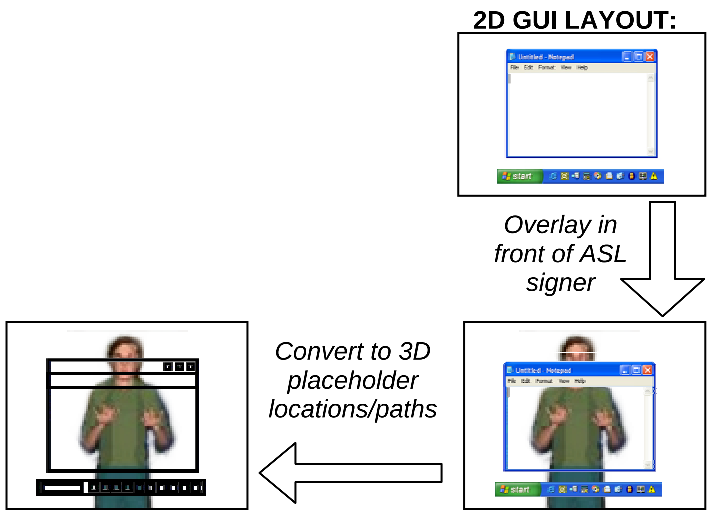

2D GUI LAYOUT:
Overlay in front of ASL signer
Convert to 3D placeholder locations/paths

Figure 8: Avoiding the Use of Scene-Visualization in GUI Applications

Representations Used in Our MT System (Part 1)

As described above, the scene-visualization software can produce a miniature "invisible world" representing the scene described by the input text. Unlike other applications of scene visualization software – where entities described by the English text would need to be rendered to the screen – in this situation, the 3D objects would be transparent. Therefore, our MT system does not care about the exact appearance of the objects being modeled. Only the location, orientation, and motion paths of these objects in some generic 3D space are important since this information will be used to produce classifier predicates for the animated ASL-signing character. This distinction between actual 3D visualizations and abstract spatial representations is captured by the use of two different data structures in our system: Visualization Scenes and Depicting Canvases.

Visualization Scene

Our ASL MT system uses a data structure called a **Visualization Scene** to represent the animation of a 3D scene that is produced by the scene-visualization software. A Visualization Scene is a record of this virtual reality animation that has been augmented with some ASL linguistic information. Specifically, a Visualization Scene is a list of **Visualized Entities**, each of which represents an individual object in the scene. A Visualized Entity includes the following information:

- A 3D Location (center of mass) of the object in the scene.
- A 3D Orientation of the object in the scene.[15]
- A pointer to some entry in the Discourse Model that is associated with the object in the scene. (The Discourse Model will be discussed later in this chapter.)

[15] In future work, Visualized Entities and Ghosts (to be discussed shortly) could store information about their size and shape (perhaps as a 3D wireframe). This information would be needed for the generation of Size and Shape Specifiers (SASSs), which are beyond the scope of this dissertation. For Classifier Predicates of Movement and Location (CPMLs), the size and shape of the depicted object is rarely important. For instance, during a CPML that conveys a collision between two objects, it is the size/shape of the ASL signer's hand that determines whether he or she has shown two objects colliding. The signer would move his or her hands together until the hands actually touch – in this case, it is not the imagined size/shape of the depicted object which determines whether a collision has been conveyed by the CPML.

Depicting Canvas

The 3D graphics information in the Visualization Scene is not used in its raw form during the CP generation process; instead, the spatial information is abstracted and some of the elements of the scene are mapped onto the region of space in front of the ASL signer (to produce the "invisible placeholders" mentioned above). A more abstract 3D scene representation called a **Depicting Canvas** will be used to record the placement of these invisible objects, their ASL-relevant features, and their mapping onto locations of space in front of the signer.

Specifically, a Depicting Canvas is a list of objects called **Ghosts**. A Ghost is an invisible placeholder in the depicting canvas which represents an object in some real-world scene being discussed. (The term "Ghost" is unique to this project.) Ghosts are not meant to be precise visual depictions of these entities; instead, they are invisible placeholders for the location and orientation of the objects in the scene. A Ghost contains the following information (much of which is similar to that of Visualized Entities):

- A 3D Location (center of mass) of the object in the scene.
- A 3D Orientation of the object.[16]
- A pointer to a Discourse Model entry corresponding to this object.

[16] This description of the information stored inside of Ghosts accounts primarily for CPMLs, not SASSs. To generate CPs discussing the structure, size, shape, and composition of objects in a scene, additional spatial information may need to be stored inside of the Ghosts. In addition to size/shape information (see Footnote 15), Ghosts may need to record information about appearance, composition, or attachment/adjacency. The precise alignment of components in a larger structure, and the degree of closeness/thickness of objects (important for non-manual signals) would also need to be modeled. Because of these complexities and others, SASSs are not part of the design focus of this dissertation.

- A set of binary features indicating whether the object is a human, an animal, a vehicle, or some other semantic category that might determine which classifier handshape should be used to refer to the object during a classifier predicate. (Multiple classifier handshapes may be permissible for the object, depending on the specific classifier predicate to be performed.)

Visualization Scenes vs. Depicting Canvases

The description above may make the Visualization Scene and Depicting Canvas models sound quite similar. While it is true that the set of features stored inside Visualized Entities and Ghosts is quite similar, there are a set of spatial transformations that the 3D scene undergoes as it is mapped from a Visualization Scene into a Depicting Canvas. A Depicting Canvas records the arrangement of objects for one particular scale and perspective of a Visualization Scene being discussed. During an ASL conversation, a signer may actually refer to the events of a scene using multiple perspectives (each involving different mappings of objects onto the signing space and different usage of classifier predicates). The scene/canvas distinction within this system is meant to account for this phenomenon. If multiple perspectives were used to describe a scene, then multiple Depicting Canvases can be created to show how to map the elements of the Visualization Scene onto the space in front of the signer.[17]

[17] Chapter 8 discusses how the use of multiple perspectives for a single scene is beyond the scope of the prototype classifier predicate generator implemented for this dissertation project.

The Depicting Canvas as a Semantic Model

The Depicting Canvas can be thought of as a partial semantic representation of a classifier predicate performance, but this semantic nature of the representation is not immediately obvious. When an ASL signer performs classifier predicates, he or she is showing how the Ghosts are arranged or move in the invisible scene. Thus, the Ghosts in the canvas and the values of their features over time are a record of the spatial information to be expressed by the classifier predicates. Of course, an MT system would also need more traditional semantic representations, but the Depicting Canvas records the 3D spatial part of the meaning. The model helps the system select the proper 3D motion paths for the signers' hands when "drawing" the 3D scenes during classifier predicates.

Canvases: A Special Kind of Model

Some of the data structures in our MT system are called **Canvases** – the Depicting Canvas, the Articulator Canvas (to be introduced later), etc. The use of the word "canvas" in the name of these models is meaningful. A canvas is a model which stores a mapping from a set of entities (that may contain more traditional linguistic features) onto the volume of space in front of the signer's torso. Thus, a canvas uses a 3D coordinate system relative to the signer's location, and it maps its entities to 3D locations in this coordinate system. The artistic implication of the term is intentional – a canvas is a data structure which records how signers "paint" their ideas onto space around them.

During an ASL conversation, the movements of a signer must be interpreted in light of the "state" of the surrounding region of space. Both the signer and the audience need to keep track of how a complex set of discourse objects and three-dimensional scene representations have been placed around the signer in order to successfully convey and understand the meaning of an utterance. In our MT system, a canvas is a linguistic model which records some of this “state” of the space surrounding the signer’s body.

Rotating, Rescaling, and Recentering the Scene

When mapping a Visualization Scene onto the volume of space in front of the signer’s body, a scale and perspective for the scene must be chosen. While the locations/orientations of the Ghosts should correspond to the location/orientation of the Visualized Entities, some details will change as the entire scene is rotated, rescaled, and recentered. While the prototype classifier predicate generator built for this dissertation project (Chapter 8) has focused on the translation of individual English sentences into animations of CPMLs, we anticipate that this mapping process should actually occur for an entire spatially-descriptive segment of English text that discusses a single 3D scene.[18] During English text analysis, collections of adjacent sentences that refer to the movement or location of a set of objects should be grouped together and sent to the scene-visualization software to produce a Visualization Scene for the entire text segment. In this case, if there are multiple sentences discussing the movement of objects in some scene, the entire set of objects throughout the group of sentences could be considered before we select a scale and perspective.

The rotation/rescaling/recentering decision may be based upon the motion path of the objects in the scene (it is easier to show some motion paths of objects if they move away

[18] While specifying a system to translate a single sentence has been a natural starting point (and a first priority), it is important to consider how this architecture would need to be enhanced to handle the translation of larger texts. If these issues are not considered early in the development process, then software design decisions may be made that would make the MT system difficult to extend.

from the signer's body), how spread-out the objects in the scene are (we would like them all to be mapped to a comfortable set of locations in front of the signer), and possibly other factors that relate to the genre of text being generated (e.g. if discussing motor vehicles, there may be guidelines for how the scene is mapped to the space in front of the signer). Chapter 8 will discuss how the implementation of a component to make this scale/perspective decision is beyond the scope of this dissertation project, but if given a set of rotation, rescaling, and recentering values, the system can calculate the location and orientation values for a Depicting Canvas from a Visualization Scene.

Generating Classifier Predicates from Ghosts

Now that the Depicting Canvas shows how to map the objects in the 3D scene onto the space in front of the signer's torso, how can our English-to-ASL MT system produce a classifier predicate performance based on this information? An overly-simplistic (and incorrect) way to produce classifier predicates would be to use a "going along for the ride" approach: When a new Ghost is introduced into the Depicting Canvas, the signing character moves its hand to a location "inside of" the transparent Ghost. By also choosing an appropriate handshape (using the +animal or +vehicle features associated with that Ghost), then a classifier predicate animation is apparently produced that conveys the spatial information from the English text. As Ghosts in the Depicting Canvas are moved or reoriented (as AnimNL analyzes more text), the signer can express this information using additional classifier predicates by again placing its hand inside the (possibly moving) 3D object.

This approach to generating classifier predicates is similar to the linguistic account of DeMatteo (1977), who sees classifier predicates as being direct "spatial analogues" of 3D movement paths in a scene imagined by the signer (Liddell, 2003b). According to DeMatteo, signers maintain a 3D mental image of a scene to be described, select appropriate handshapes to refer to entities in their model, and trace out topologically analogous location and movement paths for these entities using their hands. Unfortunately, this approach is over-generative (Liddell, 2003b). By assuming that the selection of handshapes and movements are orthogonal and that movement paths are directly representative[19] of the paths of entities in space, this analysis predicts many ASL classifier constructions that never appear in the data (containing imaginable but ungrammatical combinations of handshape, orientation, and movement) (Liddell, 2003b). Finally, this approach does not explain how to use discourse-level or non-spatial semantic information to influence the generation of classifier predicates.

Using a Lexicon of Classifier Predicate Templates

The "going along for the ride" strategy was just one way to use the 3D information in the Depicting Canvas to generate classifier predicates; this section will introduce the MT design advocated by this thesis. This design avoids the limitations of the DeMatteo approach by considering additional sources of information (from the Discourse Model and from the English semantics) during translation. We will use another linguistic account of classifier predicate generation to motivate this MT design.

[19] To illustrate how classifier predicate movements can be conventional and not visually representative, Liddell (2003b) uses the example of an upright figure walking leisurely being expressed as a classifier predicate with a "Number 1" handshape slightly bouncing as it moves along a path. While the hand bounces, the meaning is not that a human is bouncing but that he or she is walking leisurely.

Liddell (2003a, 2003b) proposed that ASL classifier predicates are stored as a large set of abstract templates in a lexicon. They are "abstract" in the sense that each template is parameterized on 3D coordinates of whatever object is being described, and each can therefore be instantiated into many possible classifier predicate outputs. For example, there may be one template for classifier predicates expressing that a car is parked at a particular 3D point in space; when this template is turned into an actual classifier predicate, then the 3D coordinate of the car would be filled in to the template.

Each lexical entry stores the semantic content of a particular classifier predicate and most of the handshape and movement specification for its performance. A signer selects a template based on how well its spatial and non-spatial semantics convey the desired content. When a signer generates a classifier predicate from this template, then the locations, orientations, and specific movement paths of objects in a 3D mental spatial representation are used to fill the remaining parameters of the template and produce a full specification of how to perform the classifier predicate.

Liddell (2003b) explains that this model avoids the under-generation of (Supalla, 1978, 1982, 1986) by incorporating a 3D spatial representation to select locations and movement paths, but it also avoids the over-generation of (DeMatteo, 1977) by restricting the possible combinations of handshapes and movement paths. Non-occurring combinations are explained as lexical gaps; ungrammatical classifier predicate feature combinations are simply not entries in the lexicon (Liddell, 2003b).

Classifier Predicate Templates and Planning

The "abstract templates with spatial-value parameters" bear a striking similarity to the Parameterized Action Representations discussed as part of the AnimNL scene visualization software. Both representations are templates whose parameters are 3D spatial information about locations, orientations, etc. A set of PARs is stored in a large database to record a collection of possible actions that can occur in an animation; during scene visualization, a template is selected from the database, its parameters are filled in, and an animation performance is produced. During classifier predicate generation in Liddell's model, a lexicon stores a collection of classifier predicate templates, a particular template is selected during generation (to express the desired semantics and spatial information), this template is filled-in, and a classifier predicate performance is produced.

We could take advantage of the above similarity by using a planning-based formalism (somewhat similar to PARs) as a basis for our linguistic encoding of classifier predicates – we could build a database of planning operator templates to serve as our lexicon of classifier predicate templates. Each planning operator template would be a specification of how to perform a particular type of classifier predicate. The classifier predicate templates would store instructions for the signing character's hand to be in a particular shape and for it move between two or more 3D coordinates in the signing space – possibly along a programmed contour. For example, we could define a planning operator that specifies how a signer's eye gaze and hands should move to show a car parking at a 3D point in space (like the example in Figure 2 on page 22).

Thus, the translation of an English sentence into a classifier predicate would actually use two different planning operator template databases with two planning processes. The first planning process would analyze the English text to produce a 3D animation of the movement of objects in a virtual reality (the Visualization Scene would be built from the English sentence). The second planning process would be used to select the proper hand and eye movements to produce a classifier predicate animation describing what happens inside this 3D virtual reality (the movements of the Ghosts in the Depicting Canvas would be used to select movements for the articulators of the signer's body). Figure 9 contains an illustration of this process.

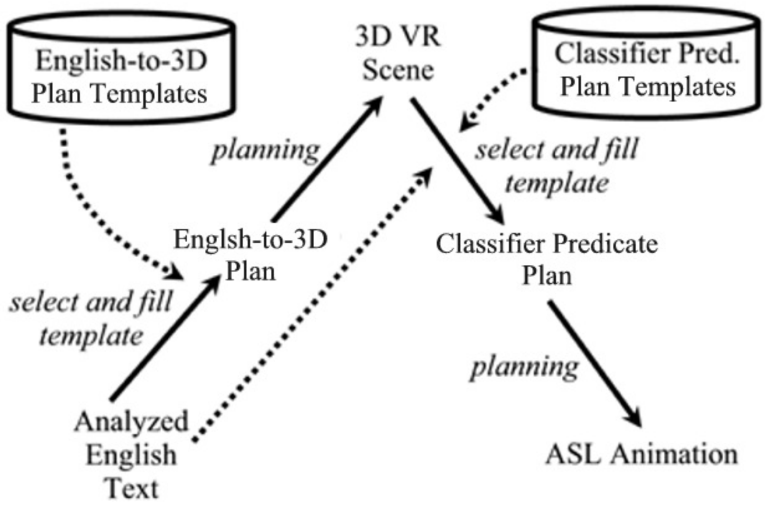


Figure 9: The Use of Two Planning Processes

PARs are actually quite complex data structures which have many sophisticated features that facilitate their use in animation planning (Badler et. al, 2000). The classifier predicate templates do not need to use the full power of the formalism; so, we have instead used a simpler planning operator formalism for writing classifier predicate templates. We will discuss the details of the format of our classifier predicate templates at the end of this chapter. While we did not actually use the full PAR formalism to encode our classifier predicate templates, it was the similarity between the Liddell template model of classifier predicates and PARs which originally inspired the planning-based approach to classifier predicate generation in this thesis.

We have now discussed four possible designs for a classifier predicate generation system (fully lexicalized, compositional rules, go-along-for-the-ride, and parameterized templates). We have identified linguistic analogies for the final three approaches, and after discussing problems with the first three approaches, we have settled on a "parameterized template" approach for our system. (Table 1 summarizes these four approaches.) After noticing a similarity between the PAR formalism and the "parameterized template" linguistic model, we have decided upon a planning-based implementation for our generation system (Huenerfauth, 2004b).

Table 1: Summary of Four Approaches to Generating Classifier Predicates

Name of Design	Fully Lexicalized	Compositional Rules	Go Along for The Ride	Parameterized Templates
Uses 3D Model?	No	No	Yes	Yes
Summary	Associate a 3D movement path with all English phrases.	Combine a set of morphemes with compositional rules.	Place hand on top of the moving object in 3D model.	Use 3D model to fill spatial parameters for CP template.
Problem	Can't List All of Them	Too Many Morphemes	Overgenerates	
Linguistic Analogy		Supalla's Polymorphemic	DeMatteo's Visual/Gestural	Liddell's Templates

Representations Used in Our MT System (Part 2)

The planning process that produces an animation of a virtual human signer performing classifier predicates uses several data structures in our MT system. A classifier predicate planning operator format has been defined, and the individual planning operators have been written to serve as classifier predicate templates. These templates access information contained in the Depicting Canvas (discussed previously) and in a Discourse Model (discussed in this section), and they produce an ASL surface-form representation that specifies the values of the animation output over time.

Discourse Model

The **Discourse Model** is a data structure used in our MT system to record discourse-level information about the set of entities currently under discussion in the ASL output. As English input text is syntactically analyzed during MT, reference-resolution algorithms could identify entities mentioned in the text, and these entities would be added to the Discourse Model. Specifically, the Discourse Model is a list of objects called **Discourse Entities**, each of which represents a particular person, object, or concept currently being discussed in the ASL output. While there are various pieces of information that may be useful to record in a discourse model, there are three features which are particularly important for the classifier predicate planning process:

- **Topicalized**: This feature is true when the entity is the current topic of the discourse. More than one entity could have this value set to true at a time. Various ASL phenomena can topicalize an entity in a conversaion: the movement of a noun phrase to a left-dislocated topic position during LS generation, performing a referring expression for an entity just prior to production of a classifier predicate discussing it, etc.

- **Identified**: This feature is true when an entity is assigned a location in the current Depicting Canvas.[20]
- **Positioned**: This feature is true when the entity is assigned a location in the Depicting Canvas, has been identified to the user, and has remained in the same location since the last time its position was communicated to the user via a classifier predicate. [21]

The classifier predicate templates consult and set the values of some of these features during the classifier predicate generation planning process. The preconditions of the classifier predicate planning operators may test the value of the predicates, and the effects of the planning operators may assign them values.

Each Discourse Entity may also record information about whether it is a human, an animal, a vehicle, etc. (If the English lexicon is augmented with this type of information for particular nouns, then this information could be added to the Discourse Entity during the syntactic analysis of the English sentence.) The Discourse Entity might also contain a pointer to the Ghost (if any) which represents this object in the Depicting Canvas.

[20] During LS generation, this feature could also be used to track if the entity has been assigned a location in the current "Token Canvas," a data structure briefly discussed in Chapter 5. (While this dissertation sketches the design of some non-classifier-predicate portions of an ASL generation system, it is the classifier predicate generator which is the focus of the implementation work for this project.)

[21] Like the "identified" feature, the "positioned" feature could be set to true for an entity via pronominals or verb agreement expression during LS generation. It is possible that locations assigned to entities during LS generation could be used as locations for those objects during subsequent CP generation. The details of this sharing of information about the use of the signing space during interleaved periods of LS and CP generation is beyond the scope of this dissertation project.

Articulator Canvas

An ASL signing performance physically consists of the coordinated movement of several parts of an ASL signer's face and body. The dominant and non-dominant hands, the eye-gaze, the head-tilt, the shoulder-tilt, and other non-manual expressions are all important elements of this performance. In the surface-form representation for this system, several of these elements of the signer's body are abstracted and recorded as a basic set of 3D objects that move through the space in front of the signer. These abstract objects will be called **Articulators**. A data structure called an **Articulator Canvas** will record the positions of the articulators during a performance.

A non-linguistic (or perhaps a phonetic) representation of an animation of a 3D virtual human character performing ASL would need to record a tremendous number of parameters over time: all of the joint angles for the face, eyes, neck, shoulders, elbows, wrists, fingers, etc. If we had to generate ASL while considering all of these values, the task would be quite difficult. The goal of an ASL phonological model is to reduce the number of independent parameters needed to be specified by the generation process while still allowing us to produce a complete animation.

Phonological models of ASL represent how the handshape, hand location, hand orientation, movement, and non-manual elements of a signing performance change over time (Padden and Perlmutter, 1987; Liddell and Johnson, 1989; Brentari, 1990; Coulter, 1993). Unfortunately, current models are not well-suited to the representation of CPMLs. They typically record hand location relative to other parts of a signer's body; so, signs like CPMLs that occur in the "neutral space" in front of the torso would require precise 3D coordinates of location in order to be modeled successfully. It is unclear how a generation process would produce the stream of 3D coordinates over time needed to specify a CPML. Hand orientation is often represented as a set of cardinal or body-relative directions (which is insufficient for the variety of hand orientations seen in CPMLs), and even those models that do specify orientation more precisely do not account for how a generation process for CPMLs could calculate this information. Finally, these models specify handshape information at a finer granularity than may be necessary for CPMLs, which typically involve a finite set of handshapes.

The surface-form representation[22] developed for this system has been designed so that it is easier to specify the performance of CPMLs. In particular, it allows for more detailed 3D motion paths and orientation settings to be recorded (which are needed for CPMLs), and it allows these values to be "copied over" (with some spatial transformations applied) from location and orientation values of Ghosts in the system's Depicting Canvas. The advantage of this approach is that it allows the surface-form representation to encode the ASL performance without having to explicitly specify 3D animation coordinates. Instead, the representation can symbolically specify which Ghost should be used as the basis for the movement of an Articulator (using a "pointer" to the appropriate object in the software system) and specify which spatial transformation function (from a finite list) should be applied to the movement path of the Ghost to create the movement of the Articulator. (Additional details about this "copying" process and spatial transformation functions will be included in the discussion in Chapter 7.)

[22] While the surface-form representation used in this ASL generation system could be interpreted as encoding information at a phonological level, we have avoided referring to the representation as a "phonological model" to avoid the appearance of making a claim about the linguistic validity of this approach. This surface-form representation of an ASL performance was designed from a natural language engineering perspective, and while it may be of interest to linguists, further study would be necessary to determine if this representation is a useful way of encoding/analyzing human-produced ASL data.

The direction at which a signer's eyes are gazing and the direction at which the head is tilted are both meaningful articulators in the production of a classifier predicate. Eye-gaze and head-tilt can be represented as a pair of 3D points in space at which these articulators are aimed. Since these points will often track a Ghost in the Depicting Canvas (or may aim at the audience), the calculation of the location value for these articulators should be straightforward. The reduction of these two articulators into a pair of 3D locations can be made because what is semantically meaningful in an ASL performance about eye-gaze and head-tilt is the point at which they are aimed, not the exact details of neck or eyeball angles. Fortunately, the animation software to be used by this system can calculate head/eye positions for a virtual character given a 3D point in space; so, this reduced model is sufficient for producing an animation.[23]

During most ASL performance (in particular during CPMLs), it is the position of the hand (not the whole arm) that is semantically meaningful. So, this surface-form representation can make another simplification. The locations in space of the dominant and non-dominant hands are recorded as another pair of 3D location coordinates. We also record for each hand the 3D orientation and which handshapes should be used. Given hand location and orientation values, there are algorithms for calculating realistic elbow/shoulder angles for a 3D virtual human character (Liu, 2003; Zhao et al., 2005); so, this concise representation is again sufficient for generating animation.

The Articulator Canvas differs slightly in its implementation from the other models in the system. All of the time-indexed linguistic models in the system (the Visualization

[23] In fact, the 3D location at which head-tilt is aimed could be somewhat approximate; it can be difficult for someone viewing an ASL signer to identify a precise 3D location at which the signer's head-tilt is aimed.

Scene, Depicting Canvas, Discourse Model, etc.) contain a list of objects of the same data type (Visualized Entities, Ghosts, Discourse Entities, etc.). The number of objects in these lists may change over time as more or fewer objects are under discussion. In contrast, the Articulator Canvas contains a fixed-membership set of Articulators that have different data types. There are three different types of articulators stored inside of the canvas: Point Articulators, Hand Articulators, and Face Articulators.

- The **Point Articulators** record a single 3D point in space in front of the signer's torso. These include the eye gaze (where it is aimed), head tilt, and shoulder tilt.
- The **Hand Articulators** record a 3D point in space where a hand is located, an orientation for the palm of the hand, and a current handshape (a particular configuration of the joint angles in the hand). There are two hand articulators: the dominant and non-dominant hands of the signer.
- The **Face Articulators** represent the other non-manual features of the signing performance. Unlike the other two types of Articulators, which can be thought of as objects that are "floating" throughout the space in front of the signer, each of these Articulators would be more like a feature structure that represents a phonological parameterization of some facial muscles conveying meaning during ASL. (The only feature of this articulator that has been implemented for the prototype system built for this dissertation is a Boolean value which indicates if the signer's eye-brows are raised. Further development is left for future work.)

Chapters 6 and 7 will discuss the way in which this system represents the timeline of ASL performance that is being constructed during generation. In general, we can think of the system as producing a time-stream of values for all of the articulators (much like the diagram in Figure 2 on page 22). Some output phenomena happen in parallel and some in sequence during the performance, and it is the job of the planning operators to build a schedule of how the values of the various articulators change during the timeline.

Classifier Predicate Planning Operators

An extended example of a classifier predicate generation process is included in Chapter 8. While this section will describe an individual template in some detail, the reader is encouraged to examine the example given in Chapter 8 for an illustration of this generation process in action.

The system contains a lexicon of classifier predicate templates that represent types of classifier predicate performances. For example, there is a template for showing the location of an animal, showing a motion path for an upright walking human, or showing how a vehicle parks relative to some other objects. Typically a template will specify a handshape to use to represent the Ghost in the Depicting Canvas. For instance, the template for animal locations uses the "Hooked V" handshape, but the template for a parking vehicle uses the "Number 3" handshape. Generally, the template does not specify the exact motion path for the classifier predicate since this value is calculated based on the particular 3D spatial location/orientation of the Ghost being described.

The classifier predicate templates are planning operators (the basic processing components of a planning algorithm); so, they also have the following fields:

- **Preconditions** are logical predicates that have to be satisfied in order for the planning operator to be added to the schedule. The planning algorithm can attempt to use other planning operators to satisfy these requirements (e.g. if you want to perform a classifier predicate about some object, then perhaps the object needs to be the current topic of conversation – if it's currently not, then we can do something else which will have the effect of topicalizing this object)

- **Actions** can come in two forms:
    - **Primitive Actions** that are the basic units of output animation that are associated with this template (it specifies how values are assigned to the Articulators for the current portion of the performance timeline).
    - **Sub-Actions** that can allow one template to break apart into a set of sub-templates to be performed in parallel or in sequence.
- **Effects** that are the semantic or discourse effects of performing this particular classifier predicate (e.g. the performance of some classifier predicates may change the state of the "topicalized," "identified," or "positioned" bits of an object in the Discourse Model)

Inside of the Actions field, values will be assigned to the locations, orientations, and handshapes of the various articulators. In this system, values for locations, orientations, and handshapes are not "static" (instantaneous) values but are "dynamic" values (which record how the value changes during some time period of the animation). For most of these planning operators, the period of time that they control is the amount of time needed to perform a single ASL classifier predicate. So, to produce an animation output, the template will need to specify a dynamic location (a movement path), a dynamic orientation (a rotation), and a dynamic handshape (how the hand changes shape during the classifier predicate) for each hand that is used during the classifier predicate. (The eye-gaze or other Point Articulators are only assigned a dynamic location value.)

Typically, we will use a location or orientation value for some Ghost in the Depicting Canvas to help us calculate the value to assign each Articulator. While we might sometimes want to exactly copy the dynamic location or dynamic orientation of a Ghost and use this for the signer's hand, we will more often wish to modify the value in some way before using it for the Articulator. We will typically apply a **spatial transformation function** to the Ghost value before assigning it to the Articulator value. A spatial transformation function tells us how to calculate new location, orientation, or handshape value from some given input location, orientation, or handshape. These functions will be discussed in more detail in Chapter 7.

Consider the example of a classifier predicate template planning operator given in Figure 10. This template will produce an animation of a signer performing a classifier predicate to show the location of an object using the handshape for stationary animals (the "Hooked V" handshape).

The "Parameters" list contains a set of variables that this planning operator will access and modify. The identifier "de0" is a variable representing some Discourse Entity in the system, and the identifier "resource_list0" is a list that contains a subset of the Articulators of the signer's body that this planning operator is allowed to control. The resource list may contain: “dom” (for the dominant hand), “ndom” (for the non-dominant hand), “eg” (for eye-gaze), “ht” (for head-tilt), or “brow” (for eye-brow raising).[24]

```
LOCATE-STATIONARY-ANIMAL

Parameters:      Discourse Entity:  de0
                 Resources List:    resource_list0

Preconditions:   (hasDom resource_list0)
                 (hasEg resource_list0)
                 (isAnimal de0)
                 (hasGhost de0)
                 (topicalized de0)
Actions:         (setLocation     target: (location dom)
                                  function: downward_arc_to_location
                                  source: (location (ghost de0)))
                 (setOrientation  target: (orientation dom)
                                  function: get_to_final_orientation
                                  source: (orientation (ghost de0))
                                  alignment: top_backpalm_front_knuckles)
                 (setHandshape    target: (handshape dom)
                                  function: get_to_final_handshape
                                  source: hooked_v_handshape)
                 (setLocation     target: (location eg)
                                  function: get_to_final_location
                                  source: (location (ghost de0)))
Effects:         (topicalized de0)
                 (identified de0)
                 (positioned de0)
```

Figure 10: An ASL Classifier Predicate Planning Operator Template

The “Preconditions” contains Boolean expressions which must be true in order for the template to be added to the plan. The preconditions “(hasDom resource_list0)” and “(hasEg resource_list0)” check that the planning operator is allowed to specify values for

[24] There are more fine-grained ways to divide the articulators of the signer’s body, but this approach works well for the prototype classifier predicate generator implemented for this project (see Chapter 8).

"dom" and "eg" – the dominant hand and eye-gaze of the signer. The precondition "(isAnimal de0)" ensures that the Discourse Entity "de0" is an animal – that is, the "isAnimal" Boolean in the "de0" object is true. The precondition "(hasGhost de0)" checks that a Ghost has been created for this entity in the Depicting Canvas. The precondition "(topicalized de0)" ensures that "de0" has already been topicalized in this discourse segment. If not, then we should do something to topicalize it, such as performing a referring expression for "de0" before the classifier predicate.

The Actions field can be used to set the values of the Articulators of the signer's body. In this example, we set the location, orientation, and handshape values for "dom" – the dominant hand. Spatial transformation functions "downward_arc_to_location," "get_to_final_orientation," and "get_to_final_handshape" are used to specify each value.

Each function takes a "source" parameter. For the location, we use "(location (ghost de0))" as our input; this is the location of the Ghost in the Depicting Canvas which corresponds to Discourse Entity "de0." Similarly, the transformation function for orientation uses "(orientation (ghost de0))" as its source. For the handshape, we use "hooked_v_handshape" as our source parameter. The identifier "hooked_v_handshape" is a constant that refers to the set of finger joint angles for the "Hooked V" handshape. The value of this constant (the numerical joint information) is stored inside a library of common ASL handshapes that is maintained by the English-to-ASL MT system.

Each of the transformation functions are calculated as follows:

- The "downward_arc_to_location" function takes a object that has a dynamic location as input, and it produces a motion path (a dynamic location output) for another object. The path that it creates is an arc-like movement that ends at the final location of its input parameter. This is meant to produce the typical downward arc-like motion path ASL signers use when they perform classifier predicates that indicate the static location of objects in a 3D scene.

- The "get_to_final_orientation" function takes an object with a dynamic orientation as input, and it produces a rotation for another object such that the second object eventually gets to the same orientation as the input object.[25]
- The "get_to_final_handshape" function takes a handshape as input, and it produces a dynamic handshape that specifies how to move the hand into the handshape that has been passed as input. In this case, it will get the hand into the "Hooked V" handshape.

The Actions field also uses a "setLocation" action to specify the location at which the signer's eye-gaze should be aimed. In this case, the value of "(location eg)" is specified using the "get_to_final_location" function, which gets the signer's eye-gaze to a goal location. In this case, we move the signer's eye-gaze to aim at the Ghost's location.

The Effects field contains Boolean expressions that should be made true at the end of this planning operator. In this example, the planning operator will set the values for the topicalized, identified, and positioned bits of the "de0" discourse model entry to true.

[25] The (setOrientation …) predicate also requires an "alignment" value, which in this example is "top_backpalm_front_knuckles." This indicates that the back of the signer's palm should be used to indicate the top of the object, and the direction of the knuckles should indicate the front of the object. Orientation alignments will be discussed in more detail in Chapter 7.

Additional Details about Planning

Chapter 8 will include an extended example of the execution of the classifier predicate planning process which will show how planning operator shown in Figure 10 is used in the system. Specifically, that chapter will describe the input to the planning process, how the planning process begins, how the set of classifier predicate templates in the system are used during planning, and how a final representation of an ASL classifier predicate performance is constructed.

Chapter 8 will also give examples of classifier predicate templates whose Action fields contain sub-Actions. The Actions field of the classifier predicate shown in Figure 10 contained a set of Primitive Actions: "setLocation," "setOrientation," and "setHandshape." This example did not show how a single template can be hierarchically decomposed into sub-templates that should be performed in parallel or in sequence.

Chapter 5: Generating Multiple Subsystems of ASL

The previous chapter shows how to produce ASL classifier predicates, but a complete ASL MT system would need to produce all forms of ASL signing – not just classifier predicates. This chapter will argue that these subsystems should differ in the resources and approaches used during generation. To accommodate these different needs, this chapter discusses how a multi-path MT architecture should be used for English-to-ASL MT. This dissertation focuses primarily on the design of an ASL classifier predicate generator (whose implementation is discussed in Chapter 8), but it is important to specify the broader design for an entire English-to-ASL MT system in order to show how the classifier predicate generation could be incorporated into a complete MT system. It is also useful to consider the broader design to ensure that representations created for the classifier predicate generator (especially for representing discourse information or the surface-form of the ASL output) could also be used in the design of the non-classifier-predicate portions of the MT architecture.

As discussed in Chapter 2, during the performance of American Sign Language, the signer can switch between two subsystems of signing, Lexical Signing (LS) and

Classifier Predicate (CP). These two subsystems of signing occupy the same articulators of the signer's body (the hands, face, eyes, shoulders, head), yet they use the space around the signer in different ways. The LS subsystem generally uses Token Space, and the CP subsystem generally uses Depicting Space.[26] (There are exceptions to this generalization. Chapter 2 discussed examples of ASL LS sentences that use a Depicting Space conceptualization of the Signing Space. Further, the selection of locations assigned to objects in one conceptualization of the signing space can affect later use of the signing space. For instance, if a signer performs some CP sentences that set up a 3D scene in the Depicting Space, the signer may chose to re-use the locations assigned to these objects when setting up a Token Space for a subsequent LS sentence.)

We have also mentioned how ASL interpreters sometimes produce Signed English when translating an English text – an English-to-ASL MT system might also need to produce Signed English output on occasion. If the system is unable to produce a CP or LS performance which conveys the meaning of an input English sentence (perhaps some part of the sentence is outside the lexical or syntactic coverage of the system), then the MT system could "fall back" on this more English-like form of signing (perhaps even merely fingerspelling portions of the original English text). While this form of output is

[26] A way for English speakers to appreciate the coexistence of the LS and CP subsystems of ASL would be to imagine the existence of some analogously distinct subsystems in English. Consider if during a communication performance, an English speaker could alternate between doing one of the following:

- speaking traditional English sentences,
- performing a meaningful system of yodeling, or
- pointing to objects in the room with their tongue.

These forms of output would all occupy the same set of mouth articulators, yet they would differ in the best way to represent their structure. They may also differ significantly in the generation process and resources that were used in their production. While the examples above are meant to be hypothetical, the final discussion chapter of this document will describe how English (and other written/spoken languages) can be conceptualized as having different subsystems or sublanguages (although none involve yodeling).

less-desirable (since it may not be understandable to deaf users of the MT system with low English literacy), it would still be better than failing to produce any output at all.

A Multi-Path Architecture for ASL

The classifier predicate generation approach in Chapter 4 is computationally expensive due to the use of 3D scene visualization software; so, we would only like to use this processing pathway when necessary. Not every English input sentence to an MT system would produce an ASL sentence containing a classifier predicate. English sentences not discussing spatial concepts would not need to be processed by the scene visualization software; in fact, most of these sentences could be successfully translated using the more traditional MT technologies of previous ASL MT systems.

For this reason, the design of our English-to-ASL MT system has multiple processing pathways (Huenerfauth, 2004a). The pathway for handling English input sentences that produce classifier predicates includes the scene visualization software, while other English input sentences undergo less sophisticated processing using a more traditional MT approach (that would be easier to implement). In this way, our CP generation component could actually be layered on top of a pre-existing English-to-ASL MT system to give it the ability to produce classifier predicates. (The development of software to handle the non-CP pathways of the system – or the use of a pre-existing MT system for this purpose – will be discussed at the end of this chapter.)

The multi-path design of this system is based upon the classic "MT Pyramid" introduced in Chapter 2 (and illustrated in Figure 6 on page 46). We will therefore begin this chapter will a discussion of the various MT architectures in the pyramid, and we will discuss whether the CP generation approach in Chapter 4 is really a form of interlingua-based MT. Finally, we will show how the design of our multi-path MT architecture is reminiscent of the "MT Pyramid."

Three Types of Machine Translation Architectures

As discussed in Chapter 3, there is an architectural spectrum along which most MT systems can be classified: direct, transfer, or interlingua (Dorr et al., 1998). When building MT systems that are higher up on the MT pyramid, there is a greater amount of domain specific development work that must be performed. While direct systems may only require a bilingual lexicon, transfer systems also require analysis and transfer rules. Interlingua-based systems require interlingua schema representations and sometimes domain-specific knowledge bases. (Constructing interlingua representations and knowledge bases for all possible translation situations is daunting if not impossible.)

Generally, in the absence of statistical or case-based information, the higher up on the pyramid that the source text is analyzed, the more complex and subtle are the divergences the system can handle. In particular, at the interlingua-processing level, a knowledge base can supplement the linguistic information, producing translations that use world knowledge and that may convey more information than was present in the source text (devoid of context). However, any of the approaches can produce a correct translation for certain inputs since not all sentences require such sophisticated analysis to be translated – some exhibit little translation divergence.[27]

[27] Statistical or case-based MT systems can partially bypass the development-work vs. translation-power tradeoff described above. (Stochastic information from parallel corpora could be incorporated into the design of direct, transfer, or possibly interlingual MT systems.) Since lexical, structural, semantic, and

As we saw in Chapter 3, non-statistical direct approaches to English-to-ASL MT generally produce simple translations that are often little more than word-to-sign dictionary look-ups. With the addition of some basic sentence reordering heuristics, such systems can occasionally produce acceptable output on simple English inputs or on those English-ASL sentence pairs that have similar word order.[28] Since no syntactic analysis is performed, there is little chance that an input sentence will be outside the linguistic coverage of the system (assuming the lexicon is of sufficient size). So, the translation process will almost always be able to produce some output. (Even if an English word is not in the translation lexicon, manual fingerspelling can be used to express the word.)

Transfer MT designs address most of the linguistic shortcomings of direct systems but do require additional linguistic resources to be developed. Chapter 3 discussed several transfer-based English-to-ASL systems (Speers, 2001; Sáfár and Marshall, 2001; Zhao et al., 2000). While these systems showed promise at handling ASL LS sentences, these systems were not able to generate ASL CP sentences. Chapter 4 illustrated how spatial 3D scene representations are required to produce classifier predicates.

Our 3D Model: Almost an Interlingua

The Visualized Scene data structure discussed in Chapter 4 (with its pointers to the Discourse Model) serves as an intermediary between the analysis of English text and the

general-knowledge information may have been used by humans during the translation of the texts in the parallel-corpus training data, then all of this information is implicitly captured in statistical systems. So, when this information is applied to a direct or transfer architecture, in effect, the system is gaining information from all of the possible levels of analysis, understanding, reasoning, and generation. Unfortunately, the difficulty of collecting and annotating corpora for a signed language means that MT designers for ASL do not have a source of empirical data for producing a statistical system.

[28] Direct systems more readily convert English text into a signing system like Signed Exact English, a manually coded form of English, not a distinct natural language, like ASL.

generation of classifier predicates in our English-to-ASL MT design. This 3D model differs from interlingua representations elsewhere in the MT literature, which resemble templates or schemata (Dorr, 1992; Dorr & Voss, 2000; Lonsdale et al., 1994). However, it does possess many of the characteristics of an interlingua representation. To see why this is the case, consider this definition of an MT interlingua:

- a typically language-neutral semantic representation that is
- useful as an intermediate representation for machine translation and
- may incorporate knowledge sources beyond the semantics of the input text.

The 3D model represents those aspects of the input text's meaning that are significant for translation to classifier predicates; thus it serves as a semantic representation within the 3D motion domain – albeit a non-traditional one due to the ontological simplicity of this domain (which has no abstract concepts, beliefs, intentions, etc.). Additionally, Chapter 4 illustrated how the 3D model is useful as an intermediate representation during MT. Finally, the ability of scene visualization software to incorporate physical constraints, collision detection, and other spatial considerations means that the 3D model uses knowledge beyond the original English text.

So, the final determinant of whether the 3D model discussed in Chapter 4 is actually an interlingua is whether the representation is language-neutral. The 3D coordinates of objects in a virtual reality model are certainly language-neutral (in fact, they don't seem linguistic at all). If the only information used during the generation of classifier predicates were these language-neutral sets of 3D locations for Visualized Entities, then the translation approach in Chapter 4 could be characterized as interlingua-based.

However, the classifier predicate planning operator templates used during generation access some information beyond these 3D features. The planning operators are triggered based upon the lexical semantics of the English motion verbs used in the original input text; in fact, each template can be matched to a predicate argument frame for a particular English verb sense. The templates also check for certain semantic properties of Discourse Entities (such as +vehicle, +human, +bulky-object, or +seated-animal) whose organization seems rather ASL specific. While our CP-generator does not truly use an interlingua, its use of use of 3D scene visualization makes it a deeper form of semantic transfer than the three transfer-based MT systems discussed in Chapter 3. Since it goes higher up the MT Pyramid than they do, we could call it a "near-interlingua."

This representation could be made more like an interlingua if the English predicate argument structures inside of the classifier predicate templates were replaced with more language-neutral semantic structures. In other words, the system's spatial semantics is already language-neutral (the 3D model), and by eliminating the use of specific English predicates inside the templates, we could make the representation of the non-spatial semantics more language-neutral as well. Instead of using an ASL-specific set of bits inside the Discourse Model (+vehicle, etc.), it may be possible to use a more language-neutral ontology of similar semantic information about each entity in the model. Because there are ways to modify this design to make it more like an interlingua, for the remainder of this document we will refer to the CP pathway of our MT design as an "interlingua" pathway (instead of the more accurate yet cumbersome "near-interlingua").

Multiple Pathways with Different Architectures

While the design of our interlingua approach to classifier predicate translation sounds promising, there is a problem. It can be difficult to build interlingua MT systems for anything but a carefully limited domain; building the linguistic and knowledge resources needed for interlingua translation on less restricted texts can entail too much overhead to be practical.

What is special about the MT problem for ASL – and the reason why an interlingua translation approach is practical – is that we can characterize and identify the “hard” input sentences, the ones that require classifier predicates for translation. These are spatially descriptive English input texts, those generally containing: spatial verbs describing locations, orientations, or movements; spatial prepositions or adverbials with concrete or animate entities; or lexical items related to other common topics or genres in which classifier predicates are typically used. Such genres (e.g. vehicle motion or furniture arrangement in a room) could be detected using the features mentioned above.

While an interlingua approach is needed to translate into classifier predicates, there are a vast number of English input sentences for which such deep analysis and reasoning would not be necessary. As discussed above, resource-lighter direct and transfer approaches can often produce a correct English-to-ASL translation from lexical or syntactic information alone.

This analysis suggests a new multi-path architecture for an MT system – one that includes a direct, a transfer, and an interlingua pathway. English input sentences within the implemented interlingua’s limited domain (the spatial location and movement concepts discussed by the CP templates) could follow that processing pathway, those sentences outside of the interlingua domain but whose syntactic features fall within the linguistic coverage of the analysis and transfer rules could use the transfer pathway, and all other sentences could use the direct pathway with its bilingual dictionary look-up. (See the diagram of the multi-path design in Figure 11.)

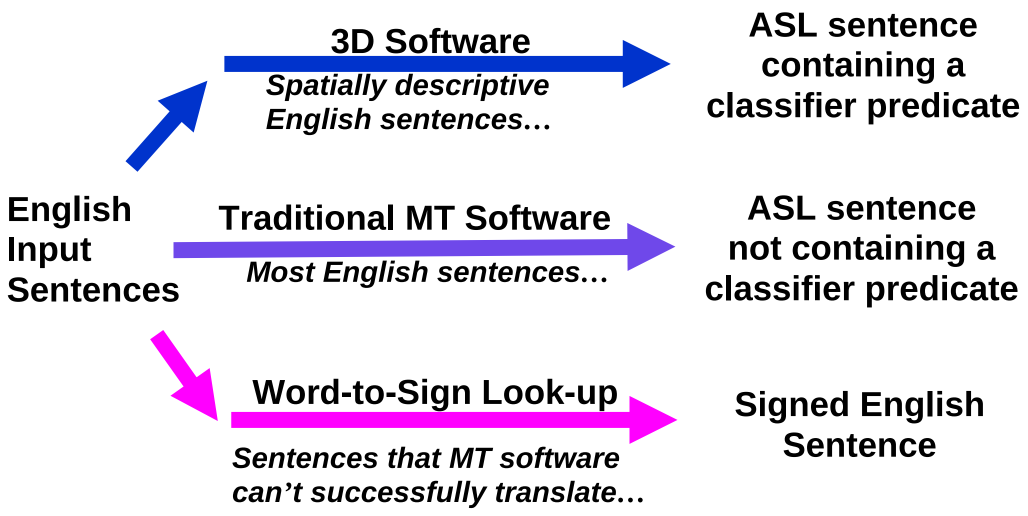

3D Software
Spatially descriptive English sentences…
ASL sentence containing a classifier predicate
English Input Sentences
Traditional MT Software
Most English sentences…
ASL sentence not containing a classifier predicate
Word-to-Sign Look-up
Sentences that MT software can't successfully translate…
Signed English Sentence

Figure 11: The Multi-Path ASL MT Architecture

Limiting the domain that the transfer and interlingua components must handle makes the development of these components more practical. The transfer pathway's analysis and transfer rules would not have to cover every English sentence: sentences whose ASL structure is English-like could use the direct pathway. Limiting domains has an even more dramatic benefit for the interlingua pathway. Instead of building complex planning-operator templates for every domain, the interlingua development can focus on the domains for which classifier predicates are used: moving people or vehicles, furniture arranged in a room, giving directions, etc. In this way, the "depth" of divergence-handling power of some translation approaches and the “breadth” of coverage of others can both be part of this multi-path architecture.

This design does more than just restrict the domains for which the interlingua must be implemented; it also reduces the ontological complexity that the entire interlingua must support. The domains listed above share a common feature: they all discuss the movement, location, orientation, and physical description of entities in three-dimensional scenes. Some phenomena whose handling often makes designing an interlingua representation quite difficult – abstract concepts, beliefs, intentions, quantification, etc. – do not need to be represented. In a sense, this multi-path architecture doesn’t just limit the things that must be represented, but the “type” of these things as well.

Choosing the Pathway: Fall-Back Architecture

Now that we have multiple processing pathways in our MT system, we must decide which pathway to use for each input sentence. We could initially implement the system as a “fall back” architecture in which the system could attempt the most complex approach (interlingua) and drop back to each of the simpler approaches whenever it lacks the proper lexical, syntactic, semantic, or knowledge resources to succeed for the current pathway. In this way, the linguistic coverage of each of the levels of representation would define exactly how input sentences would be routed through the system. Often this “fall back” will occur very quickly since only some English verbs will be associated with the kind of movement information suitable to express with a classifier predicate, or the system could decide to fall back on a simpler pathway later in the translation process. Whenever a necessary processing step cannot proceed, then we will fall back on a lower pathway. Figure 12 shows the overall design of this English-to-ASL MT system. If any step along the "interlingua" Classifier Predicate pathway fails, then we fall back on the Lexical Signing "Transfer MT Pathway." If any step on the LS pathway fails, then we fall back on the Signed English "Direct MT Pathway."

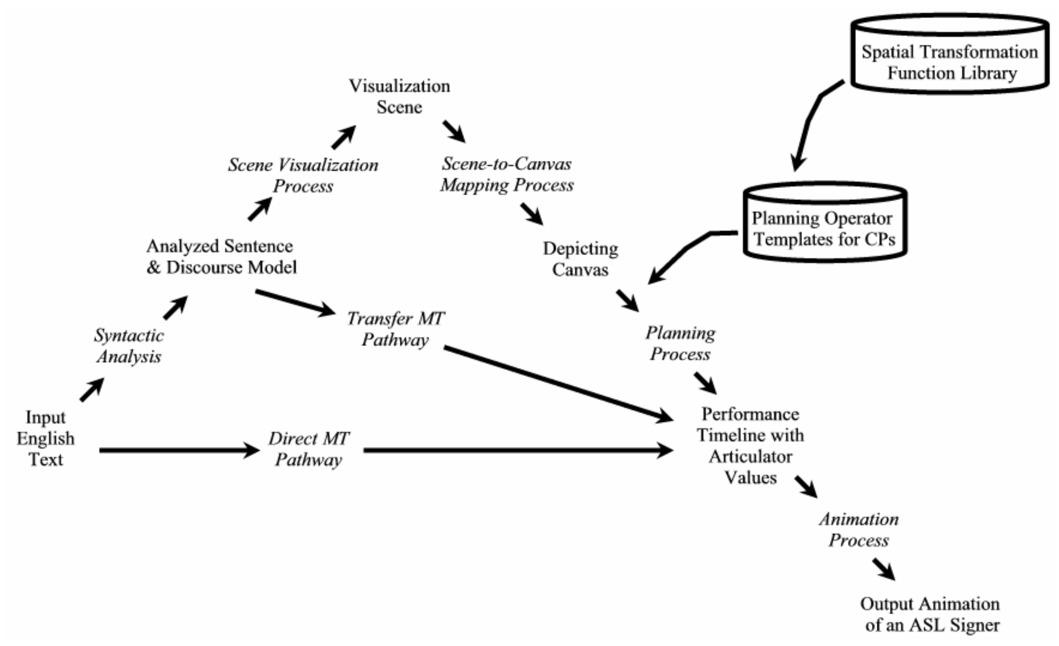

Spatial Transformation Function Library
Visualization Scene
Scene Visualization Process
Scene-to-Canvas Mapping Process
Planning Operator Templates for CPs
Analyzed Sentence & Discourse Model
Depicting Canvas
Transfer MT Pathway
Syntactic Analysis
Planning Process
Input English Text
Direct MT Pathway
Performance Timeline with Articulator Values
Animation Process
Output Animation of an ASL Signer

Figure 12: Overall English-to-ASL MT Design

In general, inside of a multi-path machine translation design, if the system were to use a more complex pathway than was necessary during translation, then, if properly implemented, output would be produced that could have been created using a simpler pathway. This is an acceptable, if less efficient, result. This situation would not always be possible in our English-to-ASL MT design since our some of our pathways do not overlap in the style of signing output they produce. Since the CP pathway produces only classifier predicates and the LS pathway produces anything but classifier predicates, these two pathways never overlap in the type of output they can produce. This situation would be possible in between the LS and SE pathways. There are situations in which an ASL LS sentence and a Signed English sentence can appear identical (for some short, simple sentences). In this case, if the system were to use the ASL LS pathway to produce an output that could have been produced with the simpler SE pathway, then the output would still be correct yet it would have been produced in a less efficient manner.

Reciprocally, if the system were to use a lower-than optimal pathway of the MT design to translate an input sentence (i.e. the system lacked the linguistic resources to translate a sentence using the sophisticated level of processing it required), then the signing output would be more English-like in structure than it should. Because many deaf users of the system would have experience using some Signed English or interacting with someone using non-fluent English-like ASL, then they may still find this overly English-like translation partially useful.

Choosing the Pathway: More Sophisticated Approaches

According to the “fall-back” design in the previous section, the system would produce a classifier predicate whenever it is possible; however, there may be situations when translating English into ASL that we should not choose to produce a classifier predicate even though it could be possible. A linguistic study of English-to-ASL translations would need to be conducted to determine how often this is the case and to understand what factors may affect this decision. The design may need to be modified to consider other factors (such as information from the surrounding sentences) to help make the decision of whether to produce a classifier predicate.

There are many factors which could complicate the decision as to whether we should use a classifier predicate to translate an English sentence into ASL:

- Some English sentences use idioms with prepositional phrases that are not meant to be interpreted spatially. For instance, "The senator is on the record as saying he will support the bill." (The senator is not sitting upon a physical "record.")
- Some English sentences are ambiguous in their meaning such that one interpretation of the sentence should be translated into an ASL sentence with a classifier predicate and one should not. For instance, "John is on the ball." (John could be sitting upon an exercise ball, or he could be someone who is well-informed and aware of his situation.)

Handling this sort of ambiguity is a common problem for machine translation systems – one which is often addressed through the use of statistical approaches which take into account features of the surrounding text and make decisions about how to translate the sentence based on information generalized from a large training corpus. While Chapter 2 discussed how there is currently insufficient parallel English-ASL corpora to develop statistical English-to-ASL MT software, it may be possible (in future work) to address the two problems listed above by taking advantage of parallel corpora of English sentences and their translation into some other written/spoken language. Specifically, if the English sentence contains a spatial preposition but the translation of the sentence into the other language does not contain spatial language, then this is a good indication that the English sentence was an idiom. Machine-learning approaches could be used to create a classifier that would identify English sentences with spatial language that are likely to be idioms. These English input sentences would not be processed using the classifier predicate pathway of the system.

Representations Must Work with Multiple Pathways

Just because we will have multiple processing architectures in our English-to-ASL MT system, that doesn't mean we should have a completely distinct set of linguistic models and representations used in each of them. Not only would it be better for engineering reasons to be able to re-use portions of the software in all three pathways, but it also makes it easier for us to merge all of these pathways at the animation output stage. In fact, we will see that our English-to-ASL MT design incorporates an ASL surface-form representation (using a multichannel timing formalism developed in Chapter 6) that is capable of representing the output of the CP (Classifier Predicate), LS (Lexical Signing), and SE (Signed English) pathways. All three pathways merge at the level of the surface-form representation, and so we only need a single animation synthesis system (to process this same format of surface-form encoding from all three pathways).

While the surface-form representation (the Articulator Canvas data structure, linked to a timeline of the ASL output being generated), could interact with all three pathways and eventually encode the output of them in a common format, the model could certainly support multiple interfaces which would allow it to interact with each pathway in a different manner. (Since this system uses an object-oriented implementation, as discussed in Chapter 7, this is particularly easy to do.) For example, lexical signing will typically involve accessing (and modifying) motion information for the hands stored in a lexicon. Initially, animation performance information would be copied from the lexicon into the surface-form representation, and then various syntactic or morphological processes may modify this information. As discussed in the previous chapter, classifier predicates specify the motion paths for the hands using a planning process that accesses information in the Depicting Canvas (often the hand mirrors the location/orientation of some Ghost in the canvas). These two methods of setting the motion values of the hands are quire different, and so the surface-form representation (the Articulator Canvas) should include methods to facilitate both forms of signing output.

While the Depicting Canvas will be used primarily by the CP subsystem and the Token Canvas (briefly discussed later in this chapter) will be primarily be used by the LS subsystem, the Discourse Model will be used by both. The previous chapter discussed how classifier predicate generation would need to access information (the identified, topicalized, and positioned bits) during planning. The Discourse Model would likely store other information about each entity that is useful during the LS generation process.

How We Could Implement the SE Pathway

While not mentioned in the above section, the MT system may also include a subsystem for producing SE (Signed English) output. This subsystem would be analogous to the LS and CP subsystems in the generator. When neither the CP nor LS generation processes can succeed, then the SE subsystem would produce English-like signing. Like the LS and CP subsystems, the output of the SE pathway would be a surface-form representation to be used by the animation software to produce an output.

It is anticipated that this portion of the English-to-ASL MT design would use a dictionary look-up process (similar to the approaches used by the Signed English systems surveyed in Chapter 3). Since the CP subsystem is the focus of this research project and since the implementation of a Signed English pathway seems like a straightforward application of previously developed technology, this pathway in the overall design will not be discussed in further detail in this dissertation.

How We Could Implement the LS Pathway

While the LS pathway is also not part of the implementation work for this dissertation project (Chapter 8), we will describe some future (post-dissertation) issues in its implementation here. In particular, some attractive properties of the overall MT design are brought to light by considering advantages that the design provides during the implementation of the LS pathway. We would likely adopt some of the technological approaches of previous English-to-ASL MT researchers (all of whom focused on the LS subsystem of the language exclusively). For example, this pathway of the system may use a transfer-based MT design that incorporates (possibly lexicalized, possibly English-synchronized) syntax rules and a lexicon of ASL signs. While Chapter 3 discussed how many of these systems had partial success at producing ASL LS animation output, that chapter also highlighted a few problems in their designs (which could more easily be addressed in the English-to-ASL MT design of this thesis).

Challenges in Implementing the LS Pathway

There are a few key linguistic challenges which must be addressed during the implementation of the LS pathway. As discussed in Chapter 3, string-based encodings of an ASL signal can introduce extraneous synchronization relationships into the signal. If this system instead used the multichannel ASL representation (in Chapter 6) to represent ASL LS output, then this issue could be addressed. (In fact, Chapter 6 includes some examples of LS sentences being encoded in the representation.)

Chapter 3 also discussed how the non-manual signals (NMS) of an ASL performance are generally encoded in insufficient detail in previous English-to-ASL MT systems. The multichannel ASL representation (in Chapter 6) could richly represent the NMS portion of the animation by including several channels that encode how a set of linguistic features (which parameterize the NMS) change over time during the performance.

The LS subsystem needs to represent how objects under discussion are associated with locations in space around the signer (using the non-topological Token Space). The 3D locations of these objects are required in order for the subsystem to successfully generate a variety of LS phenomena:

- ASL pronouns consist of a pointing-like gesture which aims at the location in space associated with the object being referred to.
- Many ASL verbs indicate (or agree with) their subjects and objects by changing their path of motion. Typically these verbs will move away from the location of their subject and toward the location of their object. In the implementation of the LS pathway, these verbs could be represented as special template-like lexicon entries whose motion paths are parameterized on the locations of their subject and object.
- While none of the previous four English-to-ASL MT systems discussed in Chapter 3 implemented it, another important form of verb agreement in ASL

involves the tilting of the signer's head toward the subject of a verb and the aiming of their eye-gaze toward the object (Neidle et al., 2000). The discussion of how eye-gaze is manipulated during the generation of classifier predicates in Chapter 4 is suggestive of how this form of head-tilt/eye-gaze agreement could be implemented in the LS subsystem.

The use of space by the LS subsystem will need some representation of how objects are associated with locations around the signer. The use of a Depicting Canvas by the classifier predicate pathway suggests how this might be done for the LS subsystem. While the Depicting Canvas records how objects called Ghosts are arranged in space (in a layout that must indicate the 3D layout of a real world scene), we could use a similar representation to record the (not necessarily topological) locations of objects during LS signing. Since the name of the object which managed the use of Depicting Space was called a Depicting Canvas, the object which manages the use of Token Space could be called a **Token Canvas**.

Token Canvas

In this design, a data structure called a Token Canvas would be used to map the layout of entities in the Discourse Model onto the signing space. Specifically, a Token Canvas records the arrangement of objects called **Tokens** in space around the signer. A Token is a data structure that records the following information:

- A 3D location for this object.
- A pointer to the entry in the Discourse Model associated with this object.

Thus, Token objects are quite similar to Ghosts, except they do not need to store a 3D orientation or information about whether they are an animal, human, vehicle, etc. Unlike Ghosts, the layout of Tokens does not need to represent the 3D layout of some real-world scene under discussion – but this does not mean that Tokens should be laid out in a totally arbitrary manner. Better arrangements of the Token Canvas do the following:

- Do not crowd the space with too many tokens.
- Position tokens for objects that will be referred to as a group near one another.
- Position tokens for objects that will be compared/contrasted on opposite sides of the token canvas.
- Position tokens for objects that will participate as multiple arguments to the same agreement-indicating verb in such a way that the expression of agreement is clear to the user. In other words, if the verb will change its motion path to indicate its subject and object, this is more clearly seen by the audience if the subject and the object are not too close together.

The way in which the system might select the best arrangement for the objects in the Token Canvas is beyond the scope of the implementation work discussed in Chapter 8 (as is the rest of the LS generation pathway), but one could imagine determining the best arrangement for a Token Canvas as a form of constraint optimization. For an entire block of English text to be translated into ASL, the system could keep track of how many items it needs to refer to with pronouns, how often objects are referred to in a collective manner, how often objects are mentioned in a comparison/contrastive manner, how often pairs of objects mentioned in the text participate as different arguments to the same predicate (whose ASL verb uses motion-path agreement). After identifying all of these constraints, the system could select an arrangement for the Tokens in the Token Canvas that best satisfies the guidelines listed in the four bullet points above.

Demonstrating Key Properties of the System

This chapter has described the design of an English-to-ASL MT system that incorporates the classifier predicate generator described in Chapter 4. While the design of the CP generator is the focus of this dissertation, it was important for this chapter to specify a broader framework for an English-to-ASL MT system in order to evaluate several of the "key properties" of the system and its design (introduced in Chapter 1). Specifically, by illustrating how the classifier predicate generator can be used within a complete English-to-ASL MT framework in this chapter, we have demonstrated how this component is ***useful for MT***.

Because of the multi-path nature of this overall English-to-ASL MT architecture, the system is also ***robust*** by design. If an English input sentence cannot be successfully processed by the classifier predicate system (because some lexical items or other semantic properties of the sentence are beyond the scope of the lexicon of classifier predicates), then the MT system would attempt to translate this sentence using its LS translation pathway[29]. If the English sentence was not discussing spatial concepts, then this would actually be correct behavior; such sentences should produce LS output in

[29] And if the LS pathway fails, then it would fall-back on its Signed English pathway, which would always produce some output – this pathway is a simple dictionary look-up process that could fingerspell any unknown words. Of course, if the user of the system has low literacy, Signed English (especially Signed English containing many fingerspelled words) would probably not be very understandable.

ASL. If the English sentence should have been translated into a classifier predicate (but the classifier predicate lexicon lacked sufficient coverage), then the system would produce an overly English-like translation of the sentence. While not optimal, this output may be partially understandable or useful to a deaf user (depending on his or her literacy skills or previous experience with English-like ASL signing or Signed English signing). Even if the precise spatial meaning of the sentence is not successfully conveyed, the deaf user would be able to gain some information from the sentence (e.g. which objects are being discussed).

Chapter 6:
A Multichannel ASL Linguistic Representation

This chapter will discuss how ASL NLG is a special form of multimodal NLG with multiple channels that should be represented linguistically. It will discuss how string-based ASL representations fail to capture coordination subtleties in an ASL performance, and it will describe a new multichannel representation of ASL linguistic structure.

The Richness of a Communication Signal

There is a tremendous amount of information contained in a typical communication signal between two people in a face-to-face conversational setting. In addition to the verbal information conveyed using speech, the two participants can convey meaningful information using changes in vocal prosody, timing, and pitch/intonation. Facial expressions, eye gazes, body posture, and manual gestures during the conversation can also substantially affect the meaning conveyed. When all of the elements of the language performance are enumerated, it is clear that a tremendous amount of simultaneous information is being sent in each direction of this communication interaction. There is certainly more information than is typically represented and processed by modern natural language processing or generation software.

Natural language generation researchers think of communication signals in a variety of ways: some as a written text string, others as a speech audio signal (with prosody, timing, volume, and intonation), and those working in Multimodal NLG as text/speech with coordinated graphics (maps, charts, diagrams, etc). For example, some Multimodal NLG researchers study how to coordinate the presentation of text information with diagrams or photographs that illustrate the concepts being described. Other Multimodal NLG researchers study "embodied conversational agents" (ECAs), computer-generated animated characters that communicate with users using speech, eye gaze, facial expression, body posture, and gestures (Cassell et al., 2000; Kopp et al., 2004).

The communication signal being sent from person to person can be thought of more formally as a data stream (a time-indexed vector of values which represent the signal at each time-slice). As described above, there is clearly a lot of information in this vector (speech, intonation, gesture, etc.). For now, we can imagine the vector as storing the audio information and visual 3D data about the appearance, location, and position of parts of the body. There are more linguistically-sophisticated ways we might choose to encode this information (discussed below), but for now it is only important to note that there is a large amount of information in this signal that must be recorded over time.

We would like some way to simplify this representation to make it more manageable to process during natural language processing or generation. Three approaches for performing this simplification (compression, filtering, partitioning) will be described in this section, and the discussion of these operations will help to motivate the ASL representation described at the end of this chapter.

Simplifying the Stream: Compression

One way to reduce the number of values in this communication stream is to select an efficient representation of the information it contains. If we know something about the regularity or structure of the signal, then we can often parameterize the signal more succinctly. For example, instead of representing the audio channel of an English speech signal as detailed pitch and amplitude values, it's more compact (and computationally expedient) to take advantage of the linguistic structure of the signal to instead represent the speech signal as a set of phonetic or phonological values which change over time. Our knowledge of the linguistics of English speech allows us to more efficiently parameterize and represent the components of the signal – thus compressing the signal into a smaller number of values over time.

To represent an ASL communication stream, audio information would not need to be encoded. One could imagine representing a signing performance in a number of different ways: as a stream of color pixels representing the individual frames of the animation, as a stream of angle values for every joint in the animated signing character's body, or as a linguistically-motivated set of values for the phonological features of each of the ASL signer's articulators. These three approaches are listed in order of how efficiently they parameterize the ASL linguistic signal (from most to fewest parameters). While all three of these representations could be used to produce a final output animation of an animated human character performing ASL, clearly we would prefer to use the more linguistically-motivated representation inside the generator. This representation would make it easier to make decisions during generation and to implement any linguistic operations that need to modify the signing performance.

Simplifying the Stream: Filtering

Another way to simplify the communication steam is to simply throw away some of the information it contains – in fact, it is the presence of a writing system for a language that allows an NLG system to do exactly this. The existence of a writing system for a language (and a community of language users who are literate in that writing system) has a major impact on the way NLG systems represent and process that language. A writing system does not transcribe all of the information in a communication signal; it significantly filters the communication stream.

Often the most important characteristic of a writing system is not what it records – it's what it does not record. For English, the writing system of the language largely omits information that a communication participant would have conveyed using volume, vocal timing, gestures, facial expressions, and eye gaze direction. Instead, the writing system primarily records the verbal information conveyed using the phonological articulators of the mouth. Some punctuation marks loosely correspond to intonation or pauses, but most prosodic information is lost. (For instance, the rising tone used by speakers to indicate a question can often be indicated in the written form using a question mark.) The writing system omits nearly all of the information conveyed by some portions of the person's body – visual channels of information (manual gestures, eye gaze direction, body posture, and facial expression) are not recorded.[30]

Literacy: People Reconstructing the Communication Stream

Since writing systems often omit large amounts of communication information that would have been present in a face-to-face communication setting, then users of the writing system must develop a literacy skill. The true power of a writing system (and of language for that matter) arises from its conventional status. For writing systems, users of the language are often trained to read and write in that particular notation, and this skill allows them to “fill in the gaps” while reading. They reconstruct the missing elements of the language signal to imagine how it would have been performed by the original speaker. (Even if readers do not actually reconstruct the facial expressions and intonation of the original speaker *per se*, they do manage to reconstruct the information that they would have gained from those parts of the communication signal – e.g. emotional state of the speaker, sarcastic tone, etc.) Thus, learning to read is more than just a process of learning to pronounce the letters of an alphabet or the words on a page. In addition to these basic memorized conventions, a trained reader can interpret and analyze a text to obtain information that has been omitted from a full communication signal.[31]

[30] Facial expression and gesture is generally not conveyed in writing, except perhaps for the occasional use of “emoticons.” ;-) Some onomatopoeic use of English orthography can be used to convey non-verbal sounds a speaker may produce during a conversation, but this capability of the writing system is also quite limited. Some texts can also convey the dialect or accent of the original speaker by using special spelling conventions, but this capability of the writing system is considered unconventional and somewhat literary.
[31] There are many other mental skills involved in the successful comprehension of the subtleties of meaning in a language performance, but the focus here is on the information gap between the written and full communication signals.

Text-Based NLG Software and the Literacy Assumption

When generating English output, NLG systems typically ignore how the facial expression, hand gestures, vocal prosody, and intonation of the output signal should be specified. Because English is a language with a formal writing system (and the target user of the NLG system is someone literate in that writing system), it can omit these other channels of output. The generation task is simplified to that of producing a written text string. The designers of NLG software are thus making a literacy assumption about their audience – namely, they are assuming that their audience will find a written text string useful (despite it being a significantly reduced form of a full communication signal).

Thus, a text-based NLG system requires literate users, to whom it can transfer some of the processing burden; they must mentally reconstruct more of the language performance than do users of a speech system or an embodied conversational agent system. Since it seems so natural to use a writing system to represent languages that have a conventionally accepted one, text-based NLG researchers do not always consider how much of the processing burden they are shifting to their users. When developing an NLG system for a language without a conventional writing system (like ASL), it's not possible to build a text-based NLG system that uses a "literacy assumption" to simplify the generation task. An ASL NLG system must produce a full animation output of a signing performance that specifies how the articulators of the signer's body are coordinated.[32]

[32] While an artificially invented ASL writing system (which filters much of the communication stream) could not be used as output, it is natural to consider whether one could be used during the internal processing of an NLG system. When the ASL output must be shown to users, then the system will need to re-synthesize a full ASL animation performance. One problem is that it is uncertain whether ASL-users would tolerate the inevitable information loss during animation synthesis from a "lossy" ASL representation. Specifically, ASL signers have less experience seeing a reduced form of their language signal. While English speakers may be able to understand the sometimes monotone output of low-quality

Simplifying the Stream: Partitioning into Channels

Since most writing systems are based on strings, text-based NLG systems can easily encode their output as a single linear output stream, namely a sequence of words/characters. The generation task is more complex for a multimodal NLG system – consider an embodied conversational agent system producing a performance of a human character with speech, gestures, facial expression, etc. The movements of each of the physically distinct articulators (body parts) of the character are responsible for changing the values across different sub-streams of the full communication signal. To simplify the generation the signal, multimodal NLG systems decompose their output into several sub-streams – we will refer to these as "channels." Dividing a communication signal into channels can make it easier to represent the various choices the generator must make; generally, a different processing component of the system will govern the output of each channel. This divide-and-conquer approach can make the communication signal easier to process. The trade-off is that these channels must be coordinated over time.

The channels of a multimodal NLG system generally correspond to natural perceptual/conceptual groupings called "modalities." Coarsely, audio and visual parts of the output are thought of as separate modalities. When parts of the output appear on different portions of the display, then they are also generally considered separate modalities. For instance, a multimodal NLG system for automobile driving directions may have separate processing channels for text, maps, other graphics, and sound effects. An Embodied Conversational Agent may have separate channels for eye gaze, facial

text-to-speech-synthesis systems, they may be relying on their literacy skills to reconstruct the missing prosodic portion of the performance. Without experience with an ASL writing system, ASL signers may not have an analogous literacy skill for "filling in" lost information during ASL animation synthesis.

expression, manual gestures, and speech audio of the animated character. Note that partitioning makes an independence assumption about how the values of the communication signal should be generated. When the channels correspond to different parts of the display or different parts of the ECA character's body, then this assumption is safer to make.

ASL Requires a Multichannel Representation

An ASL NLG system cannot collapse an entire ASL animation performance into a single-channel stream that omits large portions of the communication signal (as text-based English NLG systems do). While using a string-like representation of ASL can make it easier to use written-language NLG technologies for ASL, Chapter 3 has discussed how flattening a naturally multichannel signal into a single-channel string can introduce extraneous synchronization points in the signal. A string-like representation fails to record how some portions of the signal are not coordinated with each other during generation – it over-coordinates the performance. Since each coordination relationship that must be maintained in a signal represents an additional constraint on the animated virtual human character, representations of ASL which add unnecessary synchronization points (like the "strings with feature propagation" approach of previous systems shown in Figure 5 on page 44) make the work of the character overly difficult.

A Language Signal Spread over Multiple Channels

With no conventional string-encoding of ASL (that would compress the signal into a single stream), an ASL signal is spread over multiple channels of the output – a departure from most Multimodal NLG systems, which have a single linguistic channel/modality that is coordinated with other non-linguistic resources (Huenerfauth, 2004b). (The diagram in Figure 13 shows the channels in two multimodal systems – an ASL system and a non-ASL system – with the linguistic channels labeled in each.) Thus, an important goal of this project has been to find a representation of the hierarchical linguistic structure of information on multiple channels of ASL performance and how these structures are coordinated or non-coordinated across channels over time.

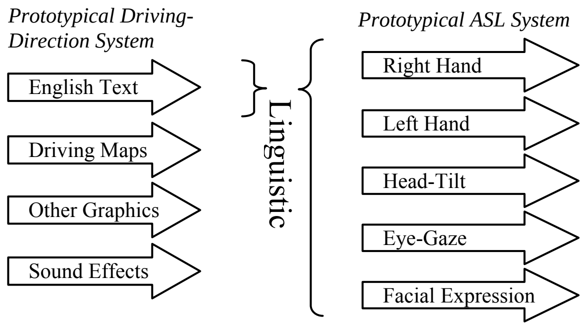


Figure 13: Linguistic Channels in Multimodal NLG Systems

Sign language animation researchers have proposed various non-hierarchical encodings of the script controlling the animated human (Elliot et al., 2004; Kennaway, 2001). While the encodings record the character's movements, they do not record the hierarchical linguistic structure of the sentence nor how larger constituents in the sentence account for phenomena across channels. They are thus not useful as a sentence-level representation during generation.

Other representations have been proposed for movements of actual humans during a performance. Signstream is a notation (and software tool) for recording parallel elements of an American Sign Language performance (Neidle et al., 2001). FORM (Martell, 2002), based on the Annotation Graph formalism (Bird and Lieberman, 2001), is a similar notation for recording non-linguistic gestures during a videotaped performance. Unfortunately, both were designed as annotation schemes, not as data structures to be used during natural language generation. They record precise timings of events; for example, they may record that someone raised an eyebrow 3.45 seconds after the start of a sentence. They don't specify which timing relationships must be coordinated and which were coincidental. They record the details of a single performance, not a set of all possible performances that would be grammatical output.

What Timing Information is Important?

An ASL communication signal contains information about several simultaneous independently-articulated channels of performance: the eye gaze, the head tilt, the shoulder tilt, the facial expression, and the location, palm orientation, and handshape of the signer's dominant and non-dominant hands. Specifying how to coordinate these channels as they change over time is an important part of ASL generation – if the temporal relationships between information on these channels are not correct, then the meaning of the ASL output could be affected. For instance, a particular facial expression may need to co-occur with a specific movement of the hands. The generator in this ASL MT system therefore represents an ASL animation performance as a set of multiple time-coordinated channels of surface-form output.

While coordination relationships are important to record, generally the precise speed of the performance does not affect its meaning. Within some reasonable limits, portions of the ASL performance could be sped up or slowed down during animation without affecting the meaning being conveyed. Therefore, the representation of an ASL signal does not necessarily need to store exact times at which events must occur. Instead, it should encode a sequence of events on each channel (and simultaneity relationships between specific events across channels of the signal).

In this ASL system, a data structure called a **Performance Timeline** will be used to manage the temporal relationships between the various Articulators in the surface-form representation. During generation, the timeline will also record how the representations of ASL syntax, semantics, and discourse change during the performance; so, these models must be linked to the performance timeline and coordinated as well. Thus, the performance timeline contains multiple parallel channels of the ASL performance. The linguistic values of each ASL articulator would be on a separate channel, and also the momentary state of each of the system's internal models of semantics, syntax, and discourse would also be listed as channels in the signal. The way in which each of these values changes during generation must be recorded over time.

Developing an ASL Representation: A Naïve Approach

Now that we understand what timing information we care about during an ASL performance (and we have seen the failings of previous string-based representations of ASL), we can develop a timing formalism tailored to the needs of ASL. The Performance Timeline object in the English-to-ASL MT system would therefore be an implementation of this timing formalism. The remainder of this chapter will describe the development of the Partition/Constitute Formalism – a mechanism for encoding coordination and non-coordination relationships across a multichannel linguistic signal

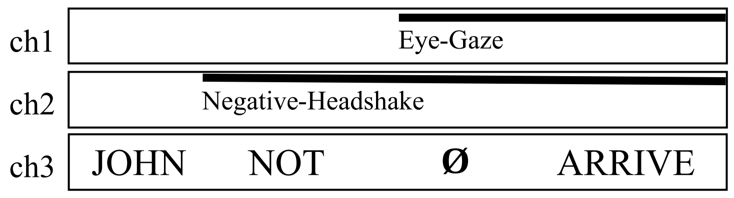
(Huenerfauth, 2005d). First, a Naïve Approach to representing ASL timing relationships will be proposed, failings of this approach will be discussed, and these failings will motivate the design of the Partition/Constitute Formalism.

A Naïve Approach: Three-Dimensional Trees

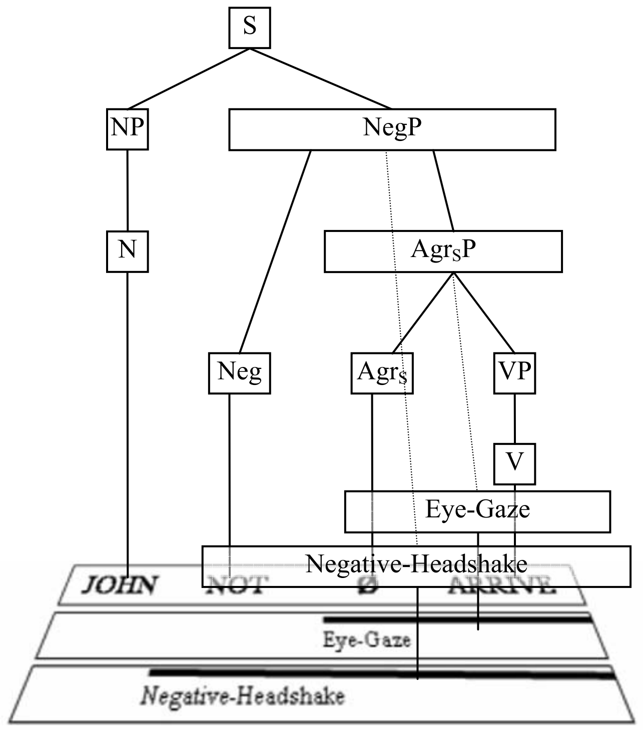
One problem we saw with the "strings with feature propagation" approaches discussed in Chapter 3, is that they broke apart multi-sign events into little pieces that were associated with each sign (thus introducing false synchronization points at every sign boundary in the system). We would like a representation that does not break apart NMS events into small artificially-divided pieces, yet we would like to link the NMS events to the hierarchical structure of the ASL sentence.

One way to do this is to represent each channel of the signal as its own string, as in Figure 14. To represent the structure of all three strings in parallel, we must use a three-dimensional tree – Figure 14(h). Some branches in the tree move out of the page toward the reader – the dotted lines in the figure that connect NegP to Negative-Headshake and that connect Agr_SP to Eye-Gaze. In fact, the nodes at the top of the tree (S, Agr_SP, NegP) are not two dimensional as they might appear. They should be visualized as 3D "blocks" that cover several channels.

(a)



(b)



(c)

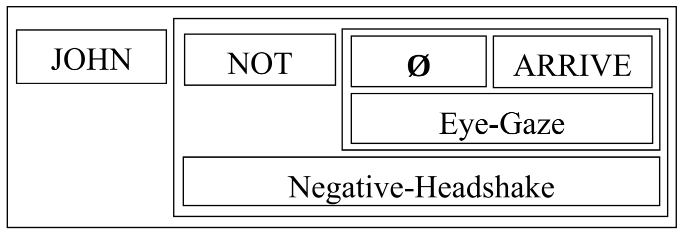


Figure 14: A Naïve Approach at Representing ASL: Three-Dimensional Trees

The tree image itself is less important than the bracketing information it captures. When viewed from above, the 3D-tree looks like the two-dimensional bracketing structure in Figure 14(c). Time is shown in the horizontal dimension, and the channels are represented in the vertical dimension. (Since they are easier to read, bracket diagrams will be used for the rest of this chapter.) The entire sentence is contained inside of a single rectangle that corresponds to the S-node in the tree. It spans the entire sentence left-to-right and specifies the sentence performance across all of the channels (top-to-bottom). To the right of the JOHN box, there is a large rectangle containing the rest of the sentence – this represents the NegP-node from the tree. When nodes covering several channels split into children, each child can cover a subset of the channels covered by their parent. For instance, the NegP node assigns its $\mathrm{Agr_S P}$ child to the top two channels and its Negative-Headshake child to the bottom channel.

There has been previous theoretical work on the definition of tree structures that can branch in multiple dimensions (Baldwin and Strawn, 1991) and grammars to generate them (Marriot and Meyer, 1996; Tucci et al., 1994). These grammars have been used to specify the structure of *visual languages* (Marriot and Meyer, 1996), a term used to refer to systematic 2D diagrams that communicate information: flow charts, state diagrams, process diagrams, etc. (While ASL is a human language that is quite visual, it is not what is meant by this term.) These grammars are multidimensional; they can produce structures that are not just linear (one-dimensional) strings. The rules in these grammars allow nodes to break into multisets of unordered sub-nodes with constraints specified between them.[33] In this chapter, we will develop a linguistically motivated two-dimensional grammar that can encode the structure of a human language signal, not just an artificial language. Our grammar uses one dimension to represent time and the other to represent the channels in a signal. This novel application of multidimensional grammar to human language (and this use of a temporal and a channel dimension) will produce a formalism that can encode the structure of a variety of multichannel language signals.

Problems with the Naïve Approach

The Naïve Approach in Figure 14 will be the basis of our new formalism; so, we carefully examine it here. Let's note what it does well. Figure 14(c) records the signal's multi-channel nature, represents how nodes break into children, and shows how responsibility for channels can be delegated to children of a node. Much like a traditional tree, the bracketing diagram breaks a signal into nested, non-overlapping components. These children nodes may divide their parent into left-to-right temporally sequential constituents and they may also assume responsibility for a subset of the top-to-bottom channels of the signal that are covered by their parent.

One problem with this approach is that it implies a sorting on the channels of the signal. Since the channels are laid out in a top-to-bottom fashion, the notation seems to imply that a total order has been defined between channels. This is not a linguistic claim that our formalism should force us to make. If the top-to-bottom layout of channels is

[33] The P/C Formalism (presented later) can be formulated as a special 2D instance of these grammars. Constituting rules could use constraints to enforce that their sub-nodes are ordered and adjacent; partitioning rules could enforce sub-nodes to cover their parent's channels in a non-overlapping way.

arbitrary, then the notation must allow non-contiguous pieces of structure to belong to a single child (Figure 15).[34]

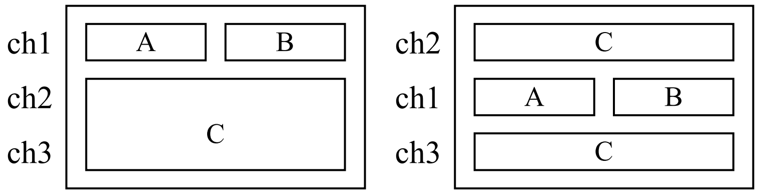


Figure 15: Equivalent Diagrams with Channels Reordered

Just because some non-contiguous structures may be needed, we don't necessarily want a formalism which will allow us to encode bizarrely-shaped nodes (i.e. nodes which represent linguistically implausible assignments of portions of the output channels to child nodes). For example, consider the unusual structures in Figure 16: these are not decompositions we need in our ASL system.

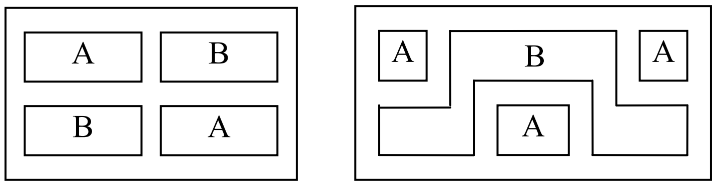


Figure 16: Linguistically Implausible Bracketing Diagrams

The bracketing structure in Figure 14(c) left some portions of the signal unspecified. (No node was assigned to some portion of some channels: consider the space below the JOHN node.) A better way to represent an unspecified part of the output signal would be

[34] Throughout the figures in this chapter, we will manage to arrange the channels in such a way as to avoid producing images that contain non-contiguous nodes; however, it is possible in this 3D tree approach for a single node to be non-contiguous in the top-to-bottom "channel" dimension of the diagram.

to use a special null node (**Ø**). In this way, when a parent branches into children nodes, the children will completely cover the range of the parent – even if we have to insert some **Ø** nodes in order to do this.[35]

Bracketing diagrams are not generally meant to indicate precise timing information (for instance, drawing a rectangle 3 millimeters to the right of another shouldn't indicate that one event happens 3 seconds after another). The diagrams are only meant to indicate linear ordering of phenomena and their nested structure. So, when a rectangle in a bracketing diagram breaks into children in both the left-to-right and top-to-bottom directions at the same time, then there may be a cross-channel temporal relationship that is left unspecified (see Figure 17).

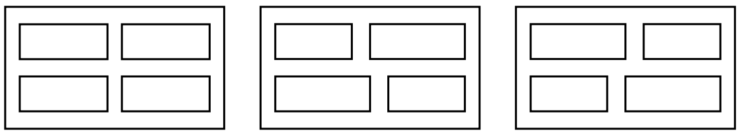


Figure 17: Should these diagrams be interpreted differently?

In this case, there are four children of the parent node: two on one channel and two on another. Unfortunately, it's not clear whether or not variations in the way we draw the diagram should be interpreted as specifying a temporal relationship between the "breaks" on each of the two channels. We can draw the break on the top channel to the left, to the right, or vertically aligned with the break on the bottom channel. Since we don't want the precise location of rectangles to indicate performance timings, then it seems awkward to

[35] This use of null nodes in the 3D trees is slightly different that the use of null symbols in linguistic syntax trees to indicate the location of a syntactic node with no lexical material. In this case, we are merely using the null nodes as structural placeholders so that our structure is a complete binary tree; we are not making a linguistic claim about the existence of a syntactic node inside the tree corresponding to this null node.

interpret the left-to-right position of these breaks as meaningful. Further, its not clear how to indicate that we don’t care about the precise timing coordination of two changes on the two different channels. Here, no matter how we draw the boxes, we seem to claim some temporal relationship.

We would prefer a multichannel representation that did not over-specify cross-channel coordination relationships (as this approach seems to do). If forced to specify temporal relationships that we don’t really care about, then we may put too many artificial requirements on the performance of the ASL animation output. Such over-specification reduces the flexibility of the final graphics animation output module of our ASL system. We may produce a specification that is too difficult for the animated human character to perform. We would prefer a formalism that allows us to *optionally* specify the coordination relationships between events on different channels – so that we only specify temporal relationships we care about – and we avoid such problems.

P/C: A Multichannel Grammar Formalism

While the Naïve Approach in Figure 14 captured the multichannel nature of the ASL signal, there were problems. It allowed us to leave portions of a diagram empty; we should use **Ø** nodes instead. We would like to avoid oddly-shaped child nodes (rectangles are preferred), but because channels may be arbitrarily ordered top-to-bottom in our diagram, we have to allow children nodes to be: *rectangles that have been sliced into horizontally parallel and identical-length pieces.* Finally, we would like an optional way to specify coordination relationships across channels of the signal.

The Naïve Approach gave us too much flexibility in the possible structures it can describe; therefore, our formalism should enforce more restrictions. We shall now require each rectangle to split in only one direction at a time. Further, we will require that the children of a rectangle “cover” all of the time (in the left-to-right dimension) and all of the channels (in the top-to-bottom dimension) of their parent in a non-overlapping way. This is our new multichannel ASL representation: The Partition/Constitute Formalism (P/C).

Since rectangles in the bracketing diagram are analogous to nodes in a 3D-tree structure, let’s replace our “rectangle” terminology with “nodes” instead. When a node branches left-to-right, we will call it a **Constituting Node**, and we say that it has broken into **Constituents**. The left-to-right ordering of constituents should be interpreted as specifying a temporal sequence for the sub-phenomena that compose their parent. Children cover the entire time range of their parent in a non-overlapping way. Constituting nodes are just like nodes in a traditional syntax tree, like Figure 3(b) on page 39, where nodes break into sequential children.

Nodes branching top-to-bottom in the bracketing diagram (i.e. branching into-the-page and out-of-the-page in the 3D tree image) are called **Partitioning Nodes**, and we say that they have broken into **Partitions**. A partitioning node breaking into children indicates a delegation of responsibility from the parent to each of its children. The set of channels covered by the parent is partitioned among all of the children in a non-overlapping manner. Each child is only allowed to specify/control those channels which it has been assigned by its parent. Since the order of channels in our diagram is arbitrary (they have been ordered in the diagram to optimize readability), then the order of a partitioning node's children in a grammar rule should not be interpreted as meaningful. Further, if a child spans multiple channels that are not adjacent in the way the diagram was drawn, then it may not appear as a single contiguous rectangle.[36]

A new P/C Formalism bracketing diagram of our sentence is shown in Figure 18; each rectangle splits left-to-right or top-to-bottom (but not both). Note that we could have also restricted the formalism so that nodes could only binary-branch. For now, we will allow multi-branching nodes, but it is interesting to note that binary-branching would imply trivially that a node can only partition or only constitute – with only two children, it could only split in one direction.

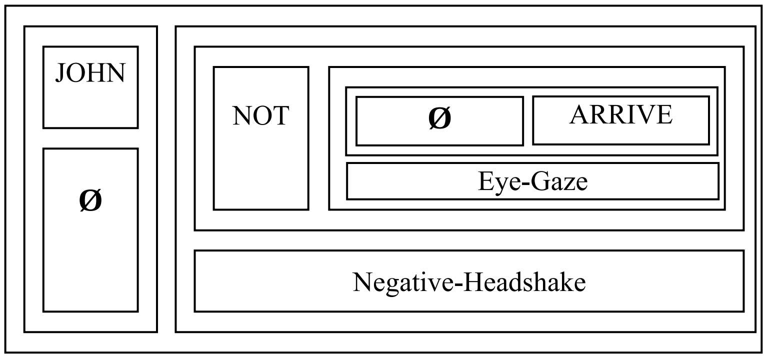


Figure 18: A Partition/Constitute Bracketing Diagram

When to Partition? When to Constitute?

When generating a representation of an ASL animation, then we can constitute or partition at any step of the derivation. How do we decide when to partition and when to constitute? Why prefer one tree to another? We could trivially partition all of the channels at the root of the tree, or we could do all of the partitioning at the leaves.

[36] As mentioned previously, we have managed to arrange the channels in the figures which appear in this chapter to avoid such non-contiguous nodes.

However, the resulting trees would not capture the conceptual decomposition of the multichannel signal that motivated this formalism. If we do all of the partitioning at the leaves, then we would produce a structure that looks just like the single-channel tree in Figure 5(b) on page 44. If we do all of the partitioning at the root, then we would have completely independent tree structures for every channel of the signal. Clearly, channels should be related in some way during a signal; so, there must be a middle ground. Constitute and partitioning nodes should be interspersed through the tree.

Guideline 1: To break a phenomenon into sub-phenomena that occur in a temporal sequence, we use a constituting node. Just like nodes in traditional syntax trees break phrases into sub-phrases, a constituting node is broken into temporally sequential children that produce their parent.

Guideline 2: If information on two different channels shows coordination in stopping/starting/intermediate timing, then we should first constitute and then partition – thereby producing a structure like that in Figure 19(a).

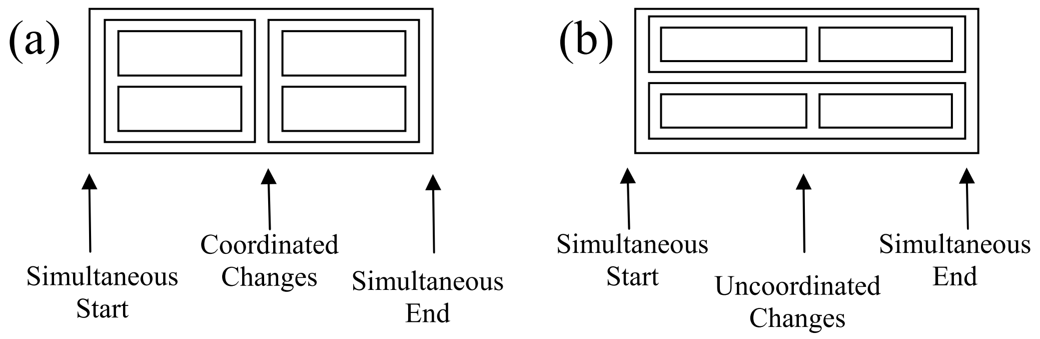


Figure 19: Specifying Coordination and Non-coordination

In Figure 19(a), the gaps in the diagram between horizontally adjacent rectangles are called **Coordination Breaks**, and they serve as cross-channel synchronization points or mileposts in the representation. Figure 19(a) represents a two-channel signal with phenomena on each channel that begin, change, and end at the same time. The coordination break captures the simultaneity between the changes in the two phenomena.

Figure 19(b) also represents a two-channel signal with phenomena that begin, change, and end; however, here the changes are not necessarily simultaneous. (The beginning and end are still coordinated.) While the "non-coordinated changes" are drawn such that they align horizontally, the diagram could have equivalently been drawn such that the boxes did not line up. Since the two channels do not have a coordination break across them, the changes may or may not align temporally during a performance. Since the relationship is not specified, the figure encodes several possible performances. We may not care to specify this relationship; it may not affect the meaning of the output.

It is important to note here that these Partition/Constitute Formalism diagrams do not indicate precise timing information. For example, one could imagine "squeezing" or "stretching" portions of the diagrams in Figure 19; these diagrams are only meant to record sequential ordering within channels and coordination relationships across channels (when there is a coordination break across those channels in the signal). During the generation of an ASL sentence in an English-to-ASL system, the precise timing of all of the events in the performance may not be initially known. While we may know some minimum/maximum timing guidelines for particular events, it will be the task of the animation output portion of the system to take a P/C encoding of an ASL sentence and to establish timings for the events on each channel that are consistent with its structure.

Guideline 3: When the timing between a signal on two channels is non-coordinated, arbitrary, or unspecified, then they should be assigned to different children of a partitioning node. Partitioning establishes a **Coordination Independence**[37] between two channels of a signal. After partitioning, the P/C tree structure does not guarantee that boundaries will align between two channels. During generation, if a portion of two channels have been assigned to different partitions, then the nodes lower in the tree structure should not need to know information (especially timing information) from nodes on a different partition to make generation choices or to produce output. By partitioning two channels before further decomposing them, we can encode that we do not require coordination between the internal structures of the two channels of the output. Thus, a syntax tree for a written string is just a special case of P/C notation that never divides a signal into partitions – the tree contains only constituting nodes. No coordination independence assumptions are made in the decomposition.

Using P/C to Represent Classifier Predicates

The examples of ASL throughout this chapter have been drawn primarily from the LS subsystem of the language. Since we would like to produce an ASL representation that can represent both LS and CP signing output, it has been important to illustrate how the P/C formalism is useful for encoding LS.[38] However, the focus of this English-to-ASL MT project is to generate animations of classifier predicates; so, we should see how

[37] The concept of coordination independence will be further clarified in Chapter 7.

[38] Building an LS component to produce a P/C representation of ASL is beyond the scope of work in Chapter 8; however, it initially appears that the syntactic transfer/generation architectures of previous ASL MT systems could be extended to produce P/C instead of strings as output. While P/C may superficially look different than traditional string-producing grammar formalisms, any traditional grammar formalism with features could be extended to produce P/C output instead. Every node in a traditional two-dimensional syntax tree can store a bit indicating whether it is a constituting or partitioning node, and we can thread a set of features throughout the entire tree representing the set of channels that are "covered" by that particular node in the tree. Thus, a traditional grammar formalism could be used to represent the constituting and partitioning operations according to the guidelines of the P/C formalism.

they can be encoded in the formalism. Classifier predicates also include many parallel elements of signing performance (eye gaze, hand locations, orientations, handshapes, etc.) that will need to be specified over time. We will see that the P/C formalism can be used to encode the coordination and non-coordination relationships between channels during classifier predicate output.

For example, the sentence “the cat sat next to the house” can be expressed using two classifier predicates (with a noun phrase preceding each). After performing the sign HOUSE, signers move their non-dominant hand in a “Spread C” handshape forward and slightly downward to a point in space in front of their torso where a miniature house is envisioned. Next, after making the ASL sign CAT, signers use their dominant hand in a “Hooked V” handshape to indicate a location where a miniature cat is envisioned. Generally, “Spread C” handshapes are used to indicate boxy objects, and “Hooked V,” stationary animals (Figure 1 on page 21). Since the sign CAT only requires one hand (and since the house is positioned relative to the CAT), a signer may choose to hold their non-dominant hand (in the “Spread C” handshape) at the house’s location during the performance of the sign CAT and the “Hooked V” classifier predicate for the cat’s location.

Figure 20 is a P/C representation of the classifier predicate performance described above. Eye-gaze, hand locations, and handshapes are represented as channels in the diagram. (There should also be an “orientation” channel for each of the hands, but these are omitted from this diagram to make it simpler for our current discussion.) The HOUSE and CAT portions of the performance are different constituents (within which the classifier predicate and the noun are sub-constituents). While the start/end of eye and hand movements should be aligned to the start/end of each classifier predicate, the P/C structure doesn't specify how these movements correspond to each other during the middle of a predicate.[39] So, they are on different partitions inside of each classifier predicate constituent. To represent how we could optionally hold the non-dominant "Spread C" hand during the CAT performance, note how the CAT part of the signal has only **Ø** nodes for the non-dominant hand. The animation output module can be designed to optionally hold the last position of a hand if the subsequent specification is only **Ø**.

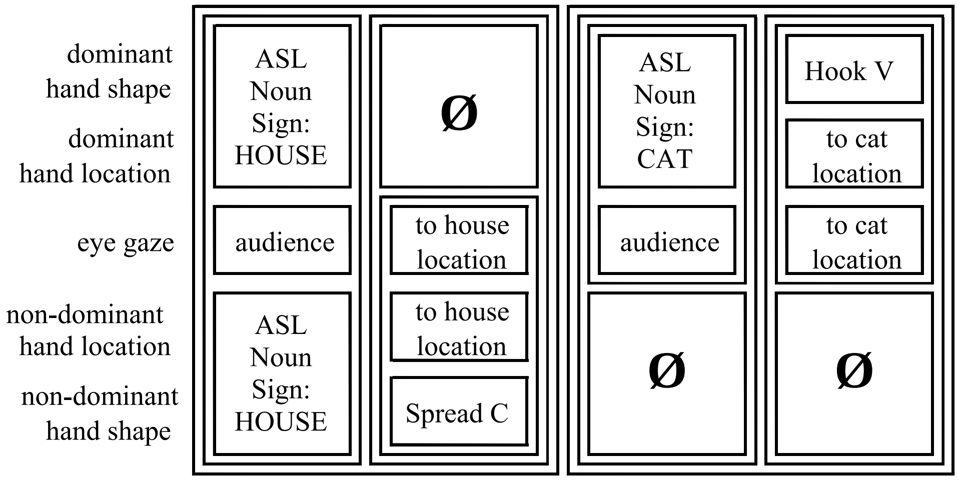


Figure 20: P/C Representation of ASL Classifier Predicates

The use of the P/C Formalism as a representation of ASL classifier predicates has an important impact on the design of the planning operator templates (as shown in Figure 10 and throughout Chapter 8). Specifically, when a planning operator divides into multiple subplans, then these subplans must be entirely sequential or entirely parallel (i.e. they

[39] The concept of coordination independence (created by partitioning nodes in P/C trees) will be clarified in Chapter 7. We will see that there are classifier predicates in which there is coordination in the movements of different parts of the body that we have shown as partitioned in the P/C diagram in Figure 20.

must constitute or must partition). In this way, the ASL representation produced by the planning process will conform to the guidelines of the P/C Formalism.

Further, when the subplans "partition" the original plan, then no two subplans must attempt to modify the values of the same articulators (they must not overlap on the same channels). Thus, as the examples of planning operators in the figures throughout Chapter 8 illustrate, each planning operator should contain a "resource list" parameter that is the set of channels of the signal it is allowed to modify. When a planning operator breaks into parallel subplans, then no two subplans can be passed the same channel of the signal. This will make it easier for a linguistic developer who is building a lexicon of classifier predicate templates to make sure that the templates conform to the "non-overlapping during partitioning" requirement of the P/C Formalism.

Comparing P/C to Gesture Representations

Earlier in this chapter, we discussed how a difference between ASL NLG and most other multimodal NLG research is that during ASL generation there are multiple channels of the signal which should be controlled in a linguistic manner. In fact, there have been some multimodal systems that have explored using linguistic structures to control (to some degree) the output of multiple channels.

Research on generating animations of an Embodied Conversational Agent that speaks and performs meaningful gestures (Cassell et al., 2000, 2001; Kopp et al., 2004; Tepper et al., 2004) has similarities to ASL generation. First of all, the channels in the signal are basically the same; an animated human-like character is shown onscreen with information about eye, face, and arm movements being generated. However, an ASL system has no audio speech channel but potentially more fine-grained channels of detailed body movement.

The less superficial similarity is that (Kopp et. al, 2004) have attempted to represent the semantic meaning of some of the character's gestures and to synchronize them with the speech output. This means that, like an ASL NLG system, several channels of the signal are being governed by the linguistic mechanisms of a natural language. Unlike ASL, the gesture system uses the speech audio channel to convey nearly all of the meaning to the user; the other channels are generally used to convey additional/redundant information. To facilitate ASL generation, the movements are broken into more fine-grained channels: the shape, orientation, and movement of each hand; the direction of head tilt, eye gaze, and shoulders; the shaking of the head; the eyebrow position; and other channels. Therefore, an ASL generator generally represents more separate channels, all of which are linguistic. So, while both systems may have multiple linguistic channels, the gesture system still has one primary linguistic channel (audio speech) and a few channels controlled in only a partially linguistic way.

The NUMACK (Kopp et al., 2004), REA (Cassell et. al, 2000), and BEAT (Cassell et al., 2001) projects all used a similar formalism to coordinate gesture and speech: a tree with nodes representing gestures that should occur during the speech output. An example of such a tree structure for the sentence "You will see Cook Hall on the right" is shown in Figure 21 – this figure is adapted from one in (Kopp et al., 2004).

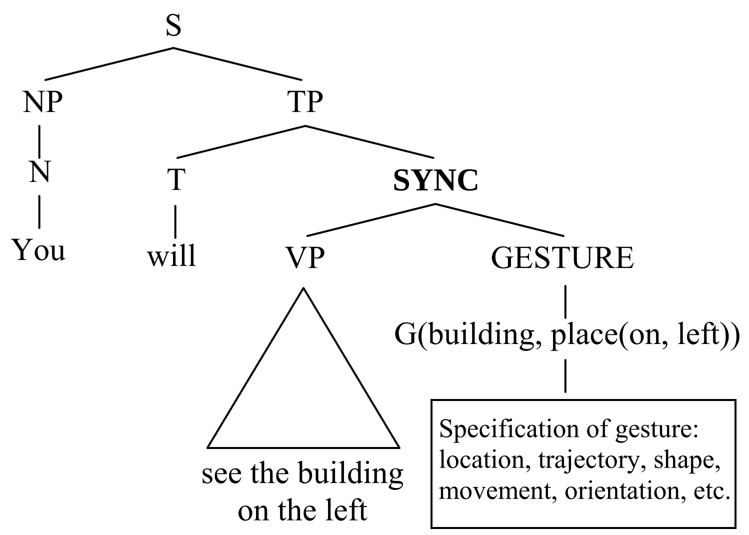

S
NP
TP
N
T
SYNC
You
will
VP
GESTURE
G(building, place(on, left))
see the building on the left
Specification of gesture: location, trajectory, shape, movement, orientation, etc.

Figure 21: Gesture/Speech SYNC Tree

While most of the branches in the tree indicate the temporal sequence of the words of the speech output, the SYNC node is special. It indicates that its two children should be performed at the same time. In this case, a gesture “G” will co-occur with the speech output of the phrase “see Cook Hall on the right.” On first glance, SYNC looks like a partitioning node in the P/C formalism. Unfortunately, SYNC trees are not expressive enough to encode a multichannel ASL animation. They do not record the internal structure of gestures; the “G” node cannot branch further into children. While some gestures may be simple enough to represent in this way, the movements needed for ASL animation are more complex. To represent the hierarchical structure of each channel, P/C thus allows partition and constituting nodes to be interleaved in a tree.

Another problem with using SYNC trees to encode ASL is that they do not model the signal channels at a sufficient level of detail. There is only a separation between the speech (the words) and gesture channels. If this were scaled up to represent the many channels of an ASL signal (by nesting SYNC nodes inside of each other or by allowing SYNC nodes to split into multiple children), then it would be difficult to ensure that no two nodes in the tree would try to modify the value of a channel at the same time. SYNC trees do not record which channels are covered by a particular node nor how responsibility for channels is delegated to children in a non-overlapping way. The P/C formalism prevents such conflicts; no two children of a partitioning node control the same channel of the signal.

While SYNC trees have proven sufficient for the generation of gestures in the REA, NUMACK, and BEAT projects, they are insufficient for ASL. The P/C formalism is more appropriate. Further, P/C may be of interest to future gesture projects that wish to produce more complex gestures (with internal structure or a more detailed decomposition of the animation channels).

Chapter 7: The Values Stored on the Linguistic Channels

The multichannel ASL representation developed in the previous chapter shows how information stored on multiple channels of a signal can be sequenced and coordinated over time. However, the chapter never discussed exactly how the information on each of these channels should be encoded in our English-to-ASL MT system. This chapter will discuss the representation of locations, orientations, handshapes, and other linguistic features (as they change in value during an ASL performance). Along the way, the chapter will highlight how a linguistic conceptualization of Ghosts, Tokens, Depicting Entities, Discourse Entities, and Articulators as bundles of features leads to a container-template-based object-oriented software implementation for this MT system.

Chapter 4 introduced the "spatial transformation functions" during the discussion of the classifier predicate planning operators. This chapter will describe these functions in more detail, discuss how they allow values to be specified inside of the ASL linguistic models, and explain how they allow us to isolate the 3D data from the rest of the ASL surface-form representation. Finally, this chapter will clarify the concept of Coordination

Independence that was originally introduced during the discussion of the Partition/Constitute formalism in Chapter 6.

The Generator and the Timeline

All of the ASL generation software in this English-to-ASL MT system is stored inside an object called the **Generator**. This generator is designed to produce a multi-channel multi-dimensional language stream that can access 3D spatial information and produce a surface-form specification according to the different language subsystems of ASL. Because of the need for multichannel coordination, this generator prominently features an object representing the ASL performance timeline. The generator also contains sets of linguistic models (Depicting Canvas, Visualization Scene, Token Canvas, Discourse Model, and Articulator Canvas) that are consulted and modified during the generation process. The values of the linguistic objects inside these models (Ghosts, Visualized Entities, Tokens, Discourse Entities, and Articulators) are tracked over time, and the values are indexed according to the Performance Timeline object below.

The **Performance Timeline** is the central coordination data structure in the generator. This object divides the timeline of an entire ASL signing performance into hierarchically-decomposed spans of time according to the Partition/Constitute Formalism discussed in the previous chapter. It is important to note the difference between the Performance Timeline and the underlying temporal semantics of the information being expressed. The Performance Timeline is a sequential record of how the ASL output is produced; this may differ significantly from the temporal organization of the events being discussed. Thus, while there are many temporal relationships that may be important to capture during a representation of event semantics, the Performance Timeline only has to record the intra-channel sequential and inter-channel coordination relationships discussed in the previous chapter.

In the diagrams of the P/C Formalism shown in the previous chapter, at the lowest level of hierarchical decomposition, there were little rectangles which "covered" a subset of the channels in the signal. Each of these little rectangles corresponds to a **Leaf** in a three-dimensional P/C tree structure. Each Leaf specifies the information on a subset of the channels of a signal during a particular time-period of the ASL performance. A Leaf encodes which channels of the ASL signal it covers/controls and it contains copies of the location, orientation, handshape, or other information that is recorded on that channel for that particular period of time.

While a Leaf represents a small period of time, it should not be thought of as a static animation snapshot. In general, a Leaf might represent a period of time of approximately 200 to 3000 milliseconds during which a single ASL lexical sign or a single classifier predicate is performed. During the linguistic part of the ASL generation process, a Leaf does not contain a specific time value; it merely represents the atomic scheduling unit of the timeline. The precise time length of each Leaf is determined by the final output animation system as it generates the performance of each sign or classifier predicate.[40]

While our discussion of the linguistic models in previous chapters may have made these data structures sound like stand-alone objects, their information is actually recorded in a distributed manner inside the Performance Timeline. For example, to look up a

[40] A Leaf can contain a "preferred" or "minimum" time duration that it should last, and this information is used by the animation when determining all of the final timing values for the performance. Chapter 8 discusses some of the details of the implementation of the animation component of the system.

value for a particular Articulator, you would navigate down to the appropriate Leaf in the Partition/Constitute tree for that ASL sentence in order to obtain the value.[41]

Linguistic Objects and Linguistic Models

Various linguistic models have been mentioned throughout this dissertation for use during classifier predicate generation (Chapter 4) or during LS generation (Chapter 5). This section will briefly summarize each of these models before suggesting how they are implemented in this English-to-ASL MT design.

- The Discourse Model is a list of Discourse Entities, which represent the objects under discussion in the text. This model keeps track of the topicalized status of these objects and other information useful during generation.
- The Visualization Scene is a list of Visualized Entities, which represent the 3D objects being discussed by spatially descriptive text. This model would typically be produced by scene-visualization software.
- The Depicting Canvas is a list of Ghosts, which represent invisible 3D placeholders that float around the signing space. The locations and orientations of these Ghosts should topologically represent the layout of some objects in a Visualization Scene.
- The Token Canvas is a list of Tokens, which represent 3D locations in space around the signer that are associated with objects under discussion. The 3D

[41] Visualized Scenes, Discourse Models, and other linguistic models are indexed on a per-sentence basis in the Performance Timeline. Only the values for the Articulator objects are recorded down at the Leaf nodes of the P/C tree for each ASL sentence.

layout of these locations is not meant to portray a 3D scene. This model would be used primarily during LS generation.

- The Articulator Canvas is a list of three different kinds of objects: two Hand Articulators, several Point Articulators, and a Face Articulator (only eye-brow raising information is specified in this dissertation). These objects represent the moving portions of the signer’s body which perform the ASL animation.

Types of Information in Each Object

The various entities differ in the set of values they store over time. Some must track the values of binary features, others need to record 3D location or orientation information, others must store pointers to entities in the discourse model, and still others need to track phonological values such as handshape or facial expression. For example, a Ghost object may need to record a pointer to a discourse model entry, a 3D location, and a 3D orientation. On the other hand, a Token object will only record a pointer to a discourse entry and a 3D location; no orientation information is needed. Information contained inside these objects is summarized in Table 2 below.

Table 2: Dynamic Values for Linguistic Objects

Object	**Dynamic Values**	**Explanation**
Ghost	Dynamic Location, Dynamic Orientation, Pointer to Discourse Entity.	Ghosts represent discourse entities with locations and orientations in the depicting canvas.
Token	Dynamic Location, Pointer to Discourse Entity.	Tokens represent discourse entities with locations in the token canvas.
Discourse Entity	Dynamic Boolean (e.g. topicalized, positioned, identified).	Discourse entities track some Boolean features used during generation.
Visualized Entity	Dynamic Location, Dynamic Orientation, Pointer to Discourse Entity.	Visualized Entities represent discourse entities modeled as 3D objects in an animation.
Point Articulator	Dynamic Location.	A point articulator must record a 3D location.
Hand Articulator	Dynamic Location, Dynamic Orientation, Dynamic Handshape.	A hand articulator must also store hand orientation and shape.

Feature Bundles and the Object-Oriented Design

By thinking of each linguistic object as a bundle of features (which is a good way to conceptualize them linguistically), then there is an obvious way to implement them in an object-oriented system. Each of the basic linguistic objects mentioned above (Ghosts, Tokens, Discourse Entities, Visualized Entities, and the different Articulators) are implemented as a class. The class stores a member which represents each of the pieces of information that are in the "feature bundle" for that type of object. So, a Token class would have at least two members: an object of type "Dynamic Location" and an object of type "Discourse Entity Pointer."

This object-oriented approach also leads to a template-based implementation of the various linguistic models (Discourse Model, Visualization Scene, Depicting Canvas, Token Canvas, etc.). Since we have stored all of the important information inside the individual linguistic objects, then the models are really just a list of the objects they contain. Thus, we don't actually need a separate class for each of the models, we can build them using an object-oriented template-class. For example, if we had an object-oriented templated class called "Container<T>" that implements whatever list-like data structure we choose to use for our models, then a Depicting Canvas would simply be an object of type "Container<Ghost>".

Basic Spatial and Linguistic Data Types

There are a small set of different data types which are recorded in the feature structures of the ASL objects listed in Table 2. Each of these value types (locations, orientations, etc.) actually has both a **Static** and **Dynamic** representation inside the generation system. The static form of the value records its momentary state during some instantaneous moment of the generation process. The dynamic form records how the value changes during the course of a single Leaf inside of a Partition/Constitute tree. Thus, the dynamic form of the value may actually store two static values (one for the start and end of the Leaf) and also some record of how the value changes over time (e.g. what 3D path was used during a change of location, etc.). Only the dynamic form of each value is actually used inside the definition of the classes listed in Table 2 above; however, often the definition of each dynamic value type will be based upon its static value definition.

Dynamic Location: Movement

Several linguistic objects need to record information about their 3D location and movement (where movement is simply a changing/dynamic location value). A **Location** is a 3D point in space (inside a particular coordinate system that is specified by the scene or canvas in which the object occurs). A **Dynamic Location** represents the change of a location value during a period of time. There are two ways to specify the value of a Dynamic Location: recording an exact 3D motion path (the scene-visualizer would do this for the Dynamic Location value of Visualized Entities) or by linking the movement of one object to that of another while applying a transformation function. (The location values for a Ghost can be specified in terms of its corresponding Visualized Entity, and the values for Articulators can be specified in terms of the Dynamic Location of a Ghost.)

Dynamic Orientation: Rotation of an Object

Some linguistic objects (Ghosts, Visualized Entities) actually represent 3D objects that have may have a distinguished front or top. Therefore, in addition to "translational" movement (moving from point X to point Y), these objects can also turn and rotate in three dimensions. An **Orientation** stores this 3D rotational positioning information by recording three rotation angles (for the x, y, and z, axis) that specify how an object is rotated from a prototypical facing-forward-sitting-upright position.[42] A **Dynamic Orientation** records how an object may rotate during a particular period of time.

[42] There are many other ways in which the orientation of an object could be specified, including through the use of a "top" and "front" direction vector, the use of quaternions, etc. In this system, the three-angle format was used in order to be compatible with the virtual human animation software used in the implementation of the classifier predicate generator (Badler et al., 2005).

Dynamic Boolean Values

A **Boolean** value simply stores either “true” or “false” – various linguistic features could be encoded using this binary approach. Sometimes the values of these binary features may change during generation. A **Dynamic Boolean** value records how a Boolean value may change during a period of time. For instance, it could stay the same, change at the start of the a Leaf in a P/C tree, or change at the end of the Leaf, etc.

Dynamic Handshapes

For the Hand Articulators, we will also need a representation which encodes how the hand should be configured (all of the joint angles). A **Handshape** represents a single static handshape, and a **Dynamic Handshape** represents a the way a handshape may change during a period of time. While many classifier predicates of movement and location used a fixed handshape throughout the performance of the predicate, there are some which require changing handshapes. For instance, some upright-human-figure classifiers will allow the index finger to wiggle during the motion path. Some vehicle classifiers can indicate that the vehicle has been damaged by slightly bending the extended fingers of the “number 3” handshape. Thus, a Dynamic Handshape object may store a changing set of Handshape values inside of itself that represent a modification of a hand over time. For the majority of Dynamic Handshape values stored in this system’s library and used during the classifier predicate generation process, the value of the Dynamic Handshape remains constant for an entire period of performance.

Spatial Transformation Functions

The example of a classifier predicate template given at the end of Chapter 4 illustrated the use of some spatial transformation functions in its specification – see Figure 10 on page 86. This figure contained primitive actions that set the values of a location, orientation, and a handshape ("setLocation," "setOrientation," and "setHandshape").[43] Each of these value-setting operators listed a "function" that should be used to calculate the value for its "target" from the value of its "source." For instance, we see in Figure 10 that the location assigned to the signer's hand (the "target") was based on the location of the Ghost object (the "source").

When we set these values, sometimes we do not just copy the value of a Ghost object – often a new value must be calculated that is based on the Ghost (but not identical to it). We would like developers with linguistic expertise (as opposed to animation expertise) to be able to produce templates of classifier predicates for this English-to-ASL MT design; so, the template itself should not contain complex animation calculations. To enable linguistic developers to produce classifier predicates whose motion paths do not exactly match those of a Ghost in the scene, a library of predefined spatial transformation functions is required. These functions are represented symbolically inside the linguistic templates during the generation process – after the output of the generator is passed off to the animation output component, then the calculations for each function are actually performed (they are finally "executed"). By providing the developer of the classifier

[43] There also exists a primitive called "setBoolean" that was not used in Figure 10; this primitive sets the value of a Dynamic Boolean.

predicate templates with this library of spatial transformation functions, the system isolates the linguistic portion of the system from the details of the 3D calculations.

Source, Implicit, and Other Parameters

Spatial transformation functions are required that can produce Dynamic Locations, Dynamic Orientations, Dynamic Handshapes, and Dynamic Boolean values as output. Generally, each type of spatial transformation function requires a “source” value that is of the same type (e.g. location, orientation, etc.) as its “target” value. For instance, the spatial transformation function “downward_arc_to_location” is used in Figure 10 to map the Dynamic Location value of a Ghost to a Dynamic Location value for the signer’s hand. In this example, the “source” is “(location (ghost de0))” – the location of the Ghost in the Depicting Canvas that corresponds to Discourse Entity “de0.”

There is an implicit parameter that is passed to all spatial transformation functions – the current value of the “target” at the beginning of this portion of the performance that is being planned. This starting-value information is often needed for spatial transformation functions that must get their “target” to some goal value. For instance the “downward_arc_to_location” function must create a motion path to a destination/goal location; in order for the function to do this, it must know where the signer’s hand (or whatever the “target” is) at the beginning of this motion path.

Aside from the “source” parameter and the implicit parameter, some spatial transformation functions require other explicit parameters to be passed to them. Later in this chapter, we will discuss the “Orientation Alignment” parameter that is required for most spatial transformation functions that set the orientation value of the signer’s hand.

We will also discuss the “Transformation Matrix” parameter that is passed to the “rotate_rescale_recenter” function.

Reusing Spatial Transformation Functions for Several CPs

During a broad-coverage implementation of a classifier predicate generator (in future work), as the number of classifier predicate templates increases, it is anticipated that the number spatial transformation functions required to implement them would “level off” – or at least grow much more slowly. A small set of such functions should be able to accommodate the spatial calculation needs of a large number of ASL classifier predicates. Chapter 8 will discuss the implementation of a prototype classifier predicate generator for this dissertation project, and one of the goals of that implementation effort was to estimate the scalability of the generator design. Measuring the growth in the number of spatial transformation functions with respect to the growth in the number of classifier predicate templates is one way to measure this scalability – and these results are presented in that chapter (see Table 4 on page 214).

Spatial Transformation Functions for Locations

For the prototype classifier predicate generator implemented for this dissertation project (see Chapter 8), there are five spatial transformation functions that produce Dynamic Location values: downward_arc_to_location, upward_arc_to_below_location, get_to_final_location, track_location, and rotate_rescale_recenter. (The “rotate_rescale_recenter” function will be explained later in this chapter.)

The “downward_arc_to_location” function produces a motion path that forms an arc-like movement that ends at the final location of its input parameter. This is meant to produce the typical downward arc-like motion path ASL signers use when they perform classifier predicates that indicate the static location of objects in a 3D scene.

The “upward_arc_to_below_location” function produces a motion path that also forms an arc-like movement; however, there are differences between this function and the previous one. The motion path produced by this function should approach the destination location from below with an upward arc-like movement, and it doesn’t quite reach this location – instead, it stops just below the destination location. This function is used in the definition of the FINAL-PLATFORM-NDOM template in Chapter 8 (see page 192).

The “get_to_final_location” function produces a movement path that arrives at the final location of its input parameter sometime before the end of the period of time it is allotted.

The “track_location” function is different than the previous three functions in that it specifies an exact location value on the very first animation frame that it is allotted. This function produces a motion path which directly copies the motion path of its input parameter.

Spatial Transformation Functions for Orientations

There are three spatial transformation functions that produce Dynamic Orientations that have been implemented for the prototype classifier predicate generator discussed in Chapter 8: get_to_final_orientation, track_orientation, and rotate_rescale_recenter. The “get_to_final_orientation” function takes an object with a dynamic orientation as input, and it produces a rotation for another object such that the second object eventually gets to the final orientation as the input object. The "track_orientation" function produces a rotation for an object such that the object's orientation over time directly copies the orientation of the input object. (The "rotate_rescale_recenter" for Dynamic Orientations will be explained later in this chapter.)

Orientation Alignment

An Orientation Alignment data structure records how we should define the "top" and "front" of the signer's hand when we map it to the orientation of some Ghost in the Depicting Canvas. We can imagine the palm of the hand as rectangular prism (which would have six sides): the inside of the palm, the back (or outside) of the palm, the side of the palm with the knuckles, the side of the palm where the wrist connects to it, the side of the palm where the thumb is, and the side of the palm where the pinky finger is. This six-sided terminology ("inside," "back," "knuckles," "wrist," "thumb," "pinky") will help us describe how the hand is oriented during a classifier predicate. For example, during the performance of a classifier predicate describing a motorized vehicle (using a "Number 3" handshape) the "thumb" side of the palm is "top" and the "knuckles" side of the palm is "front." During a classifier predicate describing an upright human figure (using a "Number 1" handshape – pictured in Figure 1 on page 21), the "knuckles" side of the palm is "top" and the "inside" of the palm is "front."

There are 24 ways we could identify a "top" and an orthogonal "front" of these six sides of the palm. For each of these 24 combinations, we can store a set of offset values for how we translate the orientation of a Ghost in a Depicting Canvas into the proper orientation for the hand which is describing it during a classifier predicate. We define each of these possible alignments as a constant and we store it in a library with names like "top_thumbside_front_knuckles" (e.g. for a motorized vehicle) or "top_knuckles_front_insidepalm" (e.g. for an upright figure). Of course, not all of these orientation alignments are actually used during ASL classifier predicates. Further, we would actually need a separate right-handed and left-handed set of orientation alignments (because the way in which the system rotates the hand is different in each case).

There is no reason why the orientation alignments that we define have to be multiples of 90 degrees – for example, during the performance of a classifier predicate for bulky objects (using the "Spread C" handshape) it is not exactly the back of the palm which indicates the top of the Ghost object being described. The palm is actually tilted downward slightly. To encode this, we can define a "bulky_object_ori_alignment" constant that encodes how to use the top/front orientation of the Ghost object in the Depicting Canvas to select a correct orientation for the signer's hand during a "bulky object" classifier predicate.

Spatial Transformation Functions for Handshapes

There are two spatial transformation functions that produce Dynamic Handshapes that have been implemented for the prototype classifier predicate generator discussed in Chapter 8: "get_to_final_handshape" and "track_handshape." The "get_to_final_handshape" function takes a handshape as input (typically a constant stored in the system's library of ASL handshapes), and it produces a Dynamic Handshape that specifies how to move the hand into the handshape that has been passed as input. The “track_handshape” function takes a Dynamic Handshape as input, and it produces a Dynamic Handshape that directly copies the handshape vales of the input over time. This function is often used to “hold” a handshape for a period of time, while the previous function is used to “get to” a handshape by the end of some period of time.

Rotate_Rescale_Recenter and the Transformation Matrix

In Chapter 4, we discussed how the system needs to rotate, rescale, and recenter the objects in the Visualization Scene so that they map onto the region of space in front of the torso of the virtual human signer. The information that specifies how to rotate, rescale, and recenter a 3D scene can be encoded as a single 4x4 transformation matrix that is used to calculate a location and orientation value for each Ghost in the Depicting Canvas using the original location and orientation values of each Visualized Entity in the Visualization Scene. (The use of a “transformation matrix” is a standard way in which computer graphics software encodes the information needed for rotating, rescaling, and recentering a 3D scene.)

There is a spatial transformation function named “rotate_rescale_recenter” that performs the mapping, and it takes two parameters: (1) a location value for a Visualized Entity and (2) a Transformation Matrix that specifies this rotation, rescaling, and recentering information. There is also a spatial transformation function for orientation values that accepts an orientation as input and returns an orientation output value.

Spatial Transformation Functions for Booleans

There is only one spatial transformation function producing Dynamic Boolean values that has been implemented for the prototype classifier predicate generator implemented in Chapter 8: “track_boolean.” This function takes a Dynamic Boolean as its input value, and it produces a Dynamic Boolean that directly copies this Boolean value over time. Typically, the only inputs that are ever passed to this function are the constants: “true_boolean” or “false_boolean.” These constants specify a Dynamic Boolean value that stays true (or stays false) for an entire period of time. By setting a Dynamic Boolean value using the “track_boolean” function with one of these constants, we are effectively maintaining the Boolean value as “true” or as “false” for an entire period of time.

Chaining Together Function Calls

Since these spatial transformation functions are not actually “executed” until the animation output stage, the linguistic generation portion of the system will “chain” function calls together. For example, let’s assume that we have a Discourse Entity “de0” that has a corresponding Visualized Entity “(visent de0)” in a Visualization Scene and a Ghost “(ghost de0)” in a Depicting Canvas. Let’s also assume that we have already calculated the 3D location and orientation values for the Visualization Scene, and they are available for us to access as “(location (visent de0))” and “(orientation (visent de0)).” Let’s trace how we would set the location value for the Ghost that corresponds to “de0” and how we would later specify a location for the signer’s dominant hand “dom” during a classifier predicate that was showing the location of this object. We will use the “rotate_rescale_recenter” function to calculate the location value “(location (ghost de0)).”

We'll need to pass the function a transformation matrix "tm0" that contains the rotation, rescaling, and recentering values. Thus, the location of the Ghost could be set as follows:

```
(setLocation    target: (location (ghost de0))
                function: rotate_rescale_recenter
                source: (location (visent de0))
                tmatrix: tm0)
```

During the classifier predicate generation process, we might specify the location value for the dominant hand "(location dom)" using another spatial transformation function (let's use the "downward_arc_to_location" function):

```
(setLocation    target: (location dom)
                function: downward_arc_to_location
                source: (location (ghost de0)) )
```

Let's replace the "setLocation" notation with something more like a traditional function notation. We will also use more concise notation for the Ghost location "g_loc," the Visualized Entity location "v_loc," and the dominant hand location "dom_loc." Using this more concise notation, we can express the two "setLocation" examples as follows:

g_loc = rotate_rescale_recenter(v_loc , tm0)

dom_loc = downward_arc_to_location(g_loc)

Since the actual 3D value of the Ghost's location coordinates may not be calculated until the final animation output stage of the system, the location value for the hand would actually be equivalent to:

dom_loc = downward_arc_to_location(rotate_rescale_recenter(v_loc , tm0))

Thus, the value of the Articulator is based on the value of a Visualized Entity with a "chain" of intermediate function applications connecting them.

A “Function” Model of ASL

Based on the discussion above, we see that there are really two ways for the system to internally store each of the dynamic spatial data types:

- The value of a Dynamic Location (or Orientation, etc) could be stored as an actual set of 3D coordinates. For instance, the scene-visualization software may produce Dynamic Location values for an object which is an exact 3D motion path for that object. The location and orientation information for Visualized Entities is typically stored in this manner, and the value of constants stored in the system’s memory (such as the finger joint information for different handshapes) is also stored in this manner.

- The value of a Dynamic Location (or Dynamic Orientation, Dynamic Handshape, etc.) could be stored as the identifier for the particular spatial transformation function that should be used to calculate its value and a list of pointers to of other objects that should be used as the inputs (i.e. “source” values) to this spatial transformation function.

This approach to representing some dynamic values in the system as functions (with a list of pointers to their inputs) actually allows the system to address a fundamental issue in ASL generation – namely, it allows us to balance two seemingly competing factors:

- The phonological specification of most human languages is categorical.

- ASL classifier predicate output often seems to be continuous or iconic.

ASL linguists are still working to resolve this categorical/continuous issue; in fact, much recent linguistic work on ASL classifier predicates has focused on this problem

(Emmorey, 2003). In this ASL design, we have worked around this problem by adopting this spatially-parameterized "function" representation of the ASL surface-form signal. As suggested by the above example of a "chain" of function calls setting the value for "dom_loc" from "v_loc," the output of this system is a complex function which is parameterized (perhaps indirectly) on the 3D values inside the Visualization Scene.

Instead of thinking about an ASL surface-form representation as a fully-defined encoding of the signal to be produced, this system treats the specification as a *composite function*. This composite function is made up of all of the individual spatial transformation functions that should be used to eventually convert the 3D values of the Visualization Entities into the values for the Ghosts and eventually into the values of the Articulators. So, the composite function is fully defined based upon a finite number of settings (the identity of the spatial transformation function to use at each step of the "chain" and the list of pointers that specify the inputs for the function at that step).

Since there will be a finite set of 3D spatial transformation functions listed in the system's resource library, then each of these functions could be uniquely identified with a finite value (and each of these functions would take a finite number of input parameters). Since there is a finite number of Articulators whose values must be specified, there would be a finite number of "chains" of function applications leading from the 3D values of the Visualized Entities to the 3D values for the Articulators.

The function accepts a non-categorical (continuous) input parameter – the 3D values of the Visualization Scene. The animation component will produce a non-categorical (continuous) output of the 3D movements of the signer's body. However, the function specification itself is categorical and finite. (See Figure 22.) Specifically, every value inside of the ASL surface-form representation records either:

- the name of a pre-defined constant from some finite list, or
- the name of a spatial transformation function and a pointer list to the values of other objects in the specification.

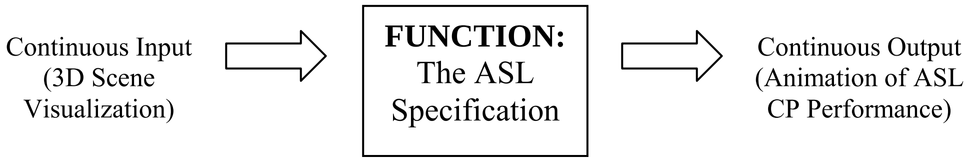

Continuous Input (3D Scene Visualization)
FUNCTION: The ASL Specification
Continuous Output (Animation of ASL CP Performance)

Figure 22: This System's "Function" Model of ASL

The ASL surface-form representation that is produced during generation (in the form of a Partition/Constitute tree structure) is a categorical representation of the animation performance that contains symbolic pointers to sources of continuous-valued 3D spatial information outside of the tree. For instance, the movement of a hand may be specified by a transformation function and a pointer to a Ghost (which is linked by a pointer to the 3D data inside of a Visualized Entity). The movement of a hand may also be specified inside of the tree by identifying an ASL sign that is being performed – the actual 3D performance data for the sign would be stored in a separate lexicon.[44]

[44] ASL signs that undergo movement path or orientation path modification (e.g. to indicate agreement) are beyond the scope of the classifier predicate generator discussed in Chapter 8; however, we could imagine how they could also fit into this "function" model. The ASL sign that the hand should perform can be specified symbolically inside of the P/C tree, and if the movement path of the needs to be modified to indicate some 3D locations or paths in the signing space, then pointers to the appropriate objects in the signing space (perhaps a Ghost or a Token) can be included in the tree structure.

Coordination Independence: A Clarification

In the definition of the Partition/Constitute Formalism in the last chapter, we said that the channels of information specified by the two subtrees below a partitioning node in a P/C tree have a coordination independence established between them. Now that we have discussed the concept of spatial transformation functions and the way in which the values of objects can be defined in terms of other objects, we can clarify this concept.

When we use the term **Coordination Independence**, what we mean is that if a surface realization component of a generation system were processing the tree structure to produce a final animation output, then it would not matter which of the two branches of the tree the system decided to process first. It is important to note that it is still possible for the timing of events on the two channels to co-occur coincidentally – the use of the "partitioning" structure in the tree merely indicates the P/C notation is not enforcing this temporal relationship itself. So, although two channels have been partitioned (and therefore have a coordination independence established between them), they could sometimes still appear coordinated.

Aside from the P/C tree structure (and aside from mere coincidence), there are other mechanisms by which the timing of the events on two channels of a signal could be made to co-occur in a coordinated manner. For instance, we've discussed how location and orientation values for an Articulator can be specified in terms of the location/orientation values of a Ghost object in the Depicting Canvas. If the channel specifying the location of the signer's eye-gaze happened to be partitioned from the channel specifying the location of the signer's hand (used to indicate the location of an object during a classifier predicate), it would still be possible for the 3D movement contour of the eye-gaze to appear that it is continually coordinated with the movement of the hand. We could define the location of both the eye-gaze and the hand in terms of the location of the same Ghost.

Thus, by defining the values of articulators on different channels of the signal in terms of the same value on deeper level of linguistic representation, a coordination relationship can be created. It is important to note that a surface realization component processing such a tree structure would still be free to calculate the output animation details of either one of the Partitioning node's subtrees first. The timing information of one subtree is not dependent on the surface-form timing information of the other subtree; instead, they are both dependent on a deeper level of linguistic representation. Thus, a coordination independence still exists between these two channels of the signal.

Chapter 8: Implementation of a Prototype CP Generator

A primary claim of this dissertation is that classifier predicate generation is an essential (yet previously-ignored) part of English-to-ASL MT and that the use of scene-visualization software, a planning-based template formalism, and a sophisticated multichannel ASL representation can allow us to build a classifier predicate generator. While Chapter 5 discussed some broader English-to-ASL MT design issues, it is the classifier predicate generator which is the focus of this project. It was important to discuss the design of the entire English-to-ASL MT system to demonstrate how the classifier predicate generator could be incorporated into a complete MT system.

Since a broad coverage implementation of a complete English-to-ASL machine translation system is well beyond the scope of this dissertation, it is important to prioritize what portion of the design should be implemented to evaluate the essential claims of this project – that it is possible to build a planning-based generator that, when given a 3D scene-visualization, is able to produce a multichannel ASL representation of a classifier predicate performance. In the English-to-ASL MT design shown in Figure 23;

it is the portion of the system that is inside of the dotted outline which is most central to this claim.

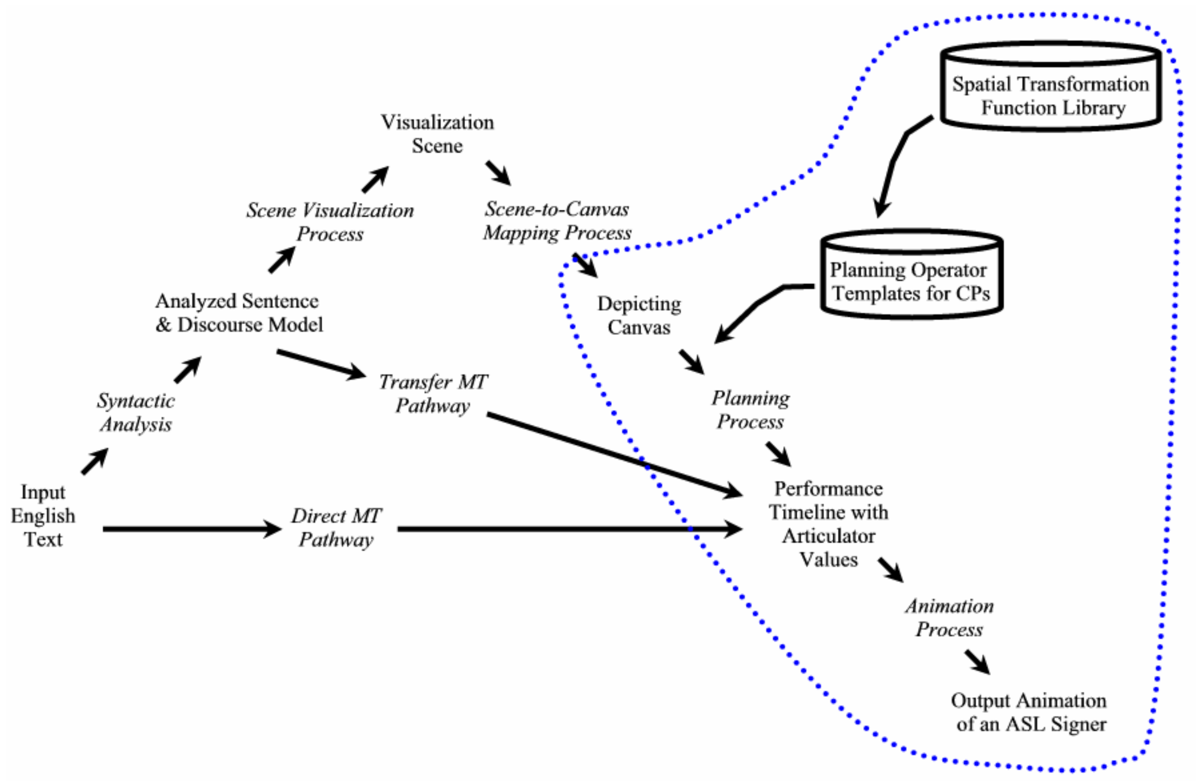

Spatial Transformation Function Library
Visualization Scene
Scene Visualization Process
Scene-to-Canvas Mapping Process
Planning Operator Templates for CPs
Analyzed Sentence & Discourse Model
Depicting Canvas
Transfer MT Pathway
Syntactic Analysis
Planning Process
Input English Text
Direct MT Pathway
Performance Timeline with Articulator Values
Animation Process
Output Animation of an ASL Signer

Figure 23: Portion of the Design Implemented

To demonstrate the functionality of this part of the design, we implemented a set of classifier predicate planning operators (Chapter 4), a planning system to processes them, a multichannel ASL surface-form representation (Chapter 6), linguistic models accessed during planning (Chapter 7), and the spatial data types (locations, orientations, etc.) and transformation functions used inside the planning operators. While the creation of a broad-coverage lexicon of classifier predicate templates would be a long-term project involving ASL linguistic developers, a small set of templates were produced so that we could evaluate the output of the system and estimate the effort needed to build a larger-scale version of the generator. (Table 5 in Chapter 9 includes a list of sentences generated by the system, which were included in a user-based evaluation study).

The output of the classifier predicate planning process is an ASL surface-form representation encoded in the Partition/Constitute Formalism; this representation specifies the values of the Articulator locations, orientations, and handshapes over time. This representation was passed to an animation system that produces a virtual human character that performs the ASL sentence. Several students[45] in the Center for Human Modeling and Simulation (HMS) modified one of the Center's virtual human characters to accept this ASL representation as its input. While the HMS students were responsible for the 3D animation programming, the specification of the requirements placed upon the animation system was part of this dissertation work – including the specification of the set of basic animation primitives that the virtual human character had to perform.

What Was Not Implemented

Several portions of the English-to-ASL MT design in Figure 23 were not implemented as part of this dissertation project. For instance, the Lexical Signing and Signed English pathways of the design have not been implemented. Further, some of the early portions of the Classifier Predicate pathway of the system make use of technologies that are not specific to ASL generation (and are the focus of other researchers):

- English Syntactic Analysis: The classifier predicate planning process requires a predicate-argument structure analysis of the English sentence to be translated.

[45] Supported by National Science Foundation Award #0520798 "SGER: Generating Animations of American Sign Language Classifier Predicates" (Universal Access Program).

Entries should also have been added to the Discourse Model for the objects mentioned in the sentence. Further, the Discourse Model entities should record whether they are human, animal, vehicle, etc. (These are tasks that are studied by English natural language understanding researchers, and they are reasonable considering the state of the art of this technology.)

- Scene Visualization: A 3D model must be produced from the English sentence description of the scene; this data is used to create a Visualization Scene which lists the objects in the scene with their locations and orientations. (This task is the research focus of the previously mentioned AnimNL (Badler et. al., 2000) and WordsEye (Coyne and Sproat, 2001) projects.)

After the English Syntactic Analysis and Scene Visualization steps are completed, the Visualization Scene must be mapped onto the space in front of the signer's torso. A scale, center point, and rotation for the 3D scene in the Visualization Scene must be selected, and this information is used to rotate and resize the scene for use as a Depicting Canvas. While this step is important for classifier predicate generation, we have chosen to focus on the implementation of the planning-based portion of the classifier predicate generation design. The selection of scale, center point, and rotation for a 3D scene will be left for future work.

To simulate the behavior of these portions of the design outlined above, a "lookup table" was created that, when given an English input sentence to be translated, returned the following information:

- A listing of the Discourse Entities that should be created for this sentence (with the appropriate information about being an animal, human, vehicle, etc.).
- A predicate-argument structure for that sentence.
- A listing of the Visualized Entities that should be created for this sentence (with their proper 3D location and orientation values).
- A scale, center point, and perspective (rotation angle) to be used during the creation of a Depicting Canvas. Thus, the mechanism which actually re-scales, recenters, and rotates the 3D scene was implemented, but the system does not need to select the scale, center point, and rotation angle on its own.

Example Set of Inputs to the CP Generator

This section will show a set of sample values stored inside of the lookup table for the English sentence: "The car parked next to the house." The first item stored inside of the lookup table is a Discourse Model created for this sentence. In this English-to-ASL MT design, the Discourse Model is encoded as a list of Discourse Entity objects. These objects store a set of predicates (stored as Boolean values) that represent important discourse information. There are three predicates (Identified, Positioned, and Topicalized) that are accessed inside of the classifier predicate generator, and values for these three predicates are stored inside of the Discourse Entities inside the lookup table. Initially, the discourse predicates **Identified** and **Positioned** would be "false" for both of the entities in the scene. Depending on the structure of the sentence, the entities may or may not be listed as **Topicalized** in the current state of the discourse model. Figure 24 contains a discourse model for this sentence. The Discourse Entity objects also records whether a Ghost has already been created for this object inside of the Depicting Canvas – this information is encoded as the “HasGhost” predicate inside of the figure. Finally, each Discourse Entity records whether the object is a vehicle, a bulky object, an upright figure, a flat object, or an animal. This information is used during the classifier predicate generation process; as new types of classifier predicates templates are added to the system (with different handshapes), additional data may need to be recorded here.

Discourse Model: a list of Discourse Entities

<car>	Topicalized:	false
	Identified:	false
	Positioned	false
	HasGhost:	false
	IsVehicle:	true
	IsBulkyObject:	false
	IsUprightFigure:	false
	IsAnimal:	false
	IsFlatObject:	false
<house>	Topicalized:	false
	Identified:	false
	Positioned:	false
	HasGhost:	false
	IsVehicle:	false
	IsBulkyObject:	true
	IsUprightFigure:	false
	IsAnimal:	false
	IsFlatObject:	false

Figure 24: Discourse Model for Extended Example

The next object stored inside of the lookup table is a Predicate Argument Structure for the English sentence. While there are various ways in which this structure could be encoded, a simple Predicate Argument Structure format has been chosen for this prototype implementation. (A more sophisticated formalism could be added to the system in future work.) See the Predicate Argument Structure in Figure 25. The structure contains a symbol that represents a particular word sense for the main verb in the sentence (in this case, "PARK-23"). This structure also stores (1) a pointer to the Discourse Entity that serves as the agent of the main verb of the sentence, (2) a string that contains the preposition from any adjunct locative phrase in the sentence, and (3) a list that contains pointers to any other Discourse Entities that are mentioned in this adjunct locative phrase. This simple encoding of the Predicate Argument Structure information is sufficient for the prototype classifier predicate generate built for this project

Predicate Argument Structure for the Sentence

Predicate:	PARK-23
Agent:	"the car", Pointer to Discourse Entity for <car>
Location Adjunct:	"next to"
NPs in Adjunct Phrase:	"the house", Pointer to Discourse Entity for <house>

Figure 25: Predicate-Argument Structure for Extended Example

We are assuming that each of the sentences loaded into this lookup table would have reached the classifier predicate generator in the multi-pathway English-to-ASL MT design described in Chapter 5. As we discussed in that chapter, the system will try to produce a classifier predicate for each input unless some step of the pathway fails. For instance, for many English input sentences, an English-to-ASL MT system may not be able to produce a Visualization Scene of the objects discussed in a text.[46] If no 3D model

[46] The system may fail in its attempt to produce a classifier predicate at any time and "fall back" on the LS pathway of the system instead. By storing a Visualization Scene in the lookup table for each of the

was successfully calculated for a span of English text, then the generator will produce LS output (i.e. without classifier predicates). If a Visualization Scene is produced for an English sentence, then the sentence would reach the beginning of the classifier predicate generation process.

Figure 26 contains a representation of the Visualization Scene for the sentence: “The car parked next to the house.” It contains an array of values for the location and orientation of the “car” and the “house” in the 3D scene. It also contains a pointer to the Discourse Model entry for each.

<u>Visualization Scene: a list of Visualized Entities</u>

<car>	Dynamic Location:	<array of 3D values>
	Dynamic Orientation:	<array of 3D values>
	Pointer to Discourse Entity for <car>.	
<house>	Dynamic Location:	<array of 3D values>
	Dynamic Orientation:	<array of 3D values>
	Pointer to Discourse Entity for <house>.	

Figure 26: Visualization Scene for Extended Example

The last piece of information stored in the lookup table for this sentence is the specification of how to rotate, rescale, and recenter the objects in the Visualization Scene to map them onto the region of space in front of the virtual human signer. As discussed in Chapter 7, the system encodes this information as a Transformation Matrix.

sentences, we are assuming that each sentence has at least made it this far in the translation process. It is not a guarantee that the system will succeed in producing a classifier predicate animation for this sentence.

Non-Planning Components of the CP Generator

As discussed in Chapter 4, the core of the classifier predicate generation system is a planning component that uses a set of planning operators to construct an ASL performance; however, there is more to the English-to-ASL MT system than this planner. These peripheral portions of the classifier predicate generator are responsible for:

- Managing the set of linguistic models (Visualization Scene, Depicting Canvas, etc.) and the spatial information they contain (Locations, Orientations, etc.).
- Storing and querying the "Lookup Table" that returns the Predicate Argument Structure, Discourse Model, Visualization Scene, and Transformation Matrix for each English sentence.
- Using the Transformation Matrix to rotate, rescale, and recenter the Visualization Scene to create a Depicting Canvas that maps the entities under discussion onto locations in space in front of the signer.
- Constructing the initial call to the planning component – establishing the "state information" inside the planning component about the current set of Discourse Entities and the Predicate Argument Structure of the English sentence we would like to translate into a classifier predicate.
- Using the output from the planning component to construct a Partition/Constitute tree structure that represents the ASL performance.
- Storing a library of ASL Handshapes (finger joint angles), common orientations (such as "upright and facing forward," needed for some classifier predicates), and

ASL signs (used to construct noun phrases that may precede a classifier predicate).[47] This data is accessed when building an animation after planning.

Creating the Depicting Canvas

After using the "Lookup Table" to get information for an English sentence, the system constructs a Depicting Canvas to encode how the objects under discussion map onto the space in front of the signer. Figure 27 shows a Depicting Canvas for the sentence: "The car parked next to the house." The location and orientation of each Ghost is specified in terms of a spatial transformation function and a pointer to a location or orientation value for a Visualized Entity. In this case, the "rotate_rescale_recenter" function also needs additional parameter: the value of the Transformation Matrix.

Depicting Canvas: a list of Ghost Objects

```
<car>          Dynamic Location:
                  rotate_rescale_recenter( Location of <car> Visualized Entity,
                                           Transformation Matrix for this Scene )
               Dynamic Orientation:
                  rotate_rescale_recenter( Orientation of <car> Visualized Entity,
                                           Transformation Matrix for this Scene )
               Pointer to Discourse Entity for <car>.

<house>        Dynamic Location:
                  rotate_rescale_recenter( Location of <house> Visualized Entity,
                                           Transformation Matrix for this Scene )
               Dynamic Orientation:
                  rotate_rescale_recenter( Orientation of <house> Visualized Entity,
                                           Transformation Matrix for this Scene )
               Pointer to Discourse Entity for <house>.
```

Figure 27: Depicting Canvas for the Extended Example

[47] The data for each ASL sign was produced by manually animating a 3D human model to perform each sign and recording the stream of joint angles for the clavicle, shoulder, elbow, wrist, and finger joints.

Planning Component of the CP Generator

Chapter 4 introduced the classifier predicate planning process, and it described the format of the planning operators that serve as classifier predicate templates (each of which contained a set of Parameters, Preconditions, Actions, and Effects). That chapter gave a high-level description of a single operator (see Figure 10 on page 86), but it did not discuss the details of the planning system that was used to process that operator. This section will describe the implementation of the planning component, discuss other planning operators (aside from those that serve as a template for a single classifier predicate), and explain how the CP generation system constructs the initial call that begins the execution of the planner.

The Hierarchical Planning Process

The planner used to process these classifier predicate planning operators is a hierarchical planner. In general, a "hierarchical planning process" begins with a list of high-level **tasks** that must be accomplished, and during the execution of the planning process, a schedule is created out of planning operator templates which accomplishes all of the tasks. As planning operator templates are selected, they may be "filled in" (if they have parameters listed, then they can have a value substituted in for their parameter).

The planner is called "hierarchical" because the initial set of tasks that are given to the planner as input can be considered "high-level" or "abstract." Some planning operators in a hierarchical planner are **methods** – they specify ways in which a high-level task can be decomposed into a set of more concrete sub-tasks. (There could be several different methods for a single task that specify alternate ways in which the task could be decomposed.) Recursively, the planner may have methods that specify how these sub-tasks could be decomposed further. Eventually, the planner will find a hierarchical decomposition of the initial set of tasks into a set of non-decomposable **primitive actions** that can be added to a schedule. This schedule of primitive actions shows us how to perform the list of high-level tasks that were initially given to the planner.

There are constraints on the planner when constructing this schedule. Specifically, when we encounter a method that has Preconditions, we have to make sure that all of the preconditions have been satisfied sometime earlier in the schedule that is being created. Thus, the planner must find a way to place the methods (i.e. the primitive actions that compose the methods) onto the schedule in such a way that the preconditions of one action have already been accomplished by the Effects of an earlier action in the schedule.

Using this hierarchical planning terminology, we can now see that the LOCATE-STATIONARY-ANIMAL classifier predicate template shown in Figure 10 is actually a "method" which is decomposed into a set of "primitive actions." In this case, the "setLocation," "setOrientation," and "setHandshape" items in the Actions field of the template are all primitive actions in the planning process. The planner does not contain methods to decompose these items further – they will be part of the final plan schedule.

Hierarchical planning begins with a list of high-level tasks given to the planner. In this classifier predicate planning system, the high-level task that starts the planning process is called "CP-GENERATE." This task takes a set of parameters that specifies an English Predicate Argument Structure that we wish to translate into a classifier predicate. There is a single method in the planning system that shows how to decompose the CP-GENERATE task. This method is shown in Figure 28. This method checks the value of the “Verb Sense Identifier” field, which contains a string that identifies the particular word sense of the main verb of the sentence. Based on this value, it selects which of the “EXPRESS-…” tasks should be used to decompose the “CP-GENERATE” task.

```
CP-GENERATE

Parameters:         Verb Sense Identifier:       string0
                    Agent Discourse Entity:      de0
                    Adjunct Preposition:         string1
                    Adjunct Discourse Entities:  de_list0
alternative 1:
    Preconditions:  (stringEqual string0 “park-23”)
    Actions:        (express-park-23    agent: de0
                                        adjunct: string1
                                        adjunctList: de_list0
                                        resources: (dom ndom eg ht brow))
alternative 2:
    Preconditions:  (stringEqual string0 “be-at-loc-34”)
    Actions:        (express-be-at-loc-34   agent: de0
                                            adjunct: string1
                                            adjunctList: de_list0
                                            resources: (dom ndom eg ht brow))

alternative 3:
    Preconditions:  (stringEqual string0 “walk-04”)
    Actions:        (express-walk-04    agent: de0
                                        adjunct: string1
                                        adjunctList: de_list0
                                        resources: (dom ndom eg ht brow))
alternative 4:
    Preconditions:  (stringEqual string0 “drive-02”)
    Actions:        (express-drive-02   agent: de0
                                        adjunct: string1
                                        adjunctList: de_list0
                                        resources: (dom ndom eg ht brow))
alternative 5: …
```

Figure 28: Code Sample from the Method for CP-GENERATE

The decomposition method for CP-GENERATE shown in Figure 28 only shows the first few “alternatives” – as additional verbs are added to the classifier predicate planning system, more alternatives could be added. Each of these alternatives would not necessarily need their own “express-…” item in their action field; verbs with similar semantics could re-use previously defined “express-…” action items. The sentence “the car parked next to the house” that has been used throughout this chapter as an extended example would match the preconditions of “alternative 1” in the method for CP-GENERATE. Later in this chapter (in the “Example Trace of the Planner Execution” section), we will continue with this example to show how this sentence would be processed by the planning system.

Implementation Details for the SHOP2 Planner

The hierarchical planner chosen for this project is the SHOP2 planner (Nau et al., 2001; Nau et al., 2003). SHOP2 is a hierarchical ordered planning system that implements the planning process described in the previous section. One feature lacking in SHOP2 is the ability to decompose a task into sub-tasks that should occur in parallel. To simulate this, all of the classifier predicate planning operators in the system contain a “Resource List” parameter (see Chapter 4). This list contains symbolic identifiers for each of the articulators of the signer’s body. If we need to write a method to decompose a task into sub-tasks that should occur in parallel, then we partition the members of the Resource List among the Resource Lists of its sub-tasks. After SHOP2 is finished planning, we use the plan schedule it produced to construct a Partition/Constitute tree that represents a classifier predicate performance. During this post-processing of the planner output, we can detect when the resource list of a task has been partitioning among the sub-tasks into which it was decomposed – when this occurs, we create a "Partition" node in the P/C tree and place each of the sub-tasks below that node.

Like most planning systems, SHOP2 allows the user to define a planning **domain** and particular planning **problems**. The file that contains the "domain" information specifies the set of methods, primitive actions, and first-order predicate logic rules that are used to perform planning for any list of tasks given to the planner. For the CP generator, all of the classifier predicate planning operators (including the LOCATE-STATIONARY-ANIMAL method shown in Figure 10 and the CP-GENERATE method shown in Figure 28) are stored inside of the domain file. This information is the same for every sentence that we ask the classifier predicate generator to produce.

The "problem" file contains the information that is specific to a particular sentence that we are trying to generate. This file contains the truth values for a set of Boolean predicates that are used during the planning process, and it contains the initial list of high-level tasks that begin the planning process. For our classifier predicate generator, the problem file is produced at run-time whenever the system needs to generate a classifier predicate. The file contains predicates that encode information about the state of the Discourse Entities at the start of the planning process: whether or not they are topicalized, whether they are animals, whether they are vehicles, etc. This information should be sent to the planner for every Discourse Entity that is used inside of the sentence that we are trying to translate into a classifier predicate.

Setting up the Surrounding Objects

After the list of predicates about the Discourse entities has been constructed, we must create the initial list of high-level tasks to pass to the planner. The final item on this list is the CP-GENERATE task for the particular sentence that we are trying to translate into a classifier predicate; it is preceded by a list of OPTIONAL-POSITION-ENTITY tasks. During a classifier predicate performance that shows the movement or location of some object, it is often important to first indicate the location of the other surrounding objects in the scene. In particular, when performing a classifier predicate that shows the location of an object relative to some other objects in the sentence, it is required that these other objects have first been located in space. For example, in the sentence "The car parked between the house and the cat" whose ASL classifier predicate translation was shown in Figure 2 on page 21, the signer showed the location of the house and the cat before performing the classifier predicate for the parking car.

To produce this behavior in our classifier predicate generator, a task called OPTIONAL-POSITION-ENTITY is added to the initial task list for each Discourse Entity in the Discourse Model of the sentence being generated. The method which decomposes the OPTIONAL-POSITION-ENTITY task is shown in Figure 29.

```
OPTIONAL-POSITION-ENTITY

Parameters:          Discourse Entity:          de0

alternative 1:
    Actions:         <empty>

alternative 2:
    Preconditions:   (one-entity-has-been-positioned)
    Actions:         (position-entity-dom      entity: de0
                                               resources: (dom ndom eg ht brow))

alternative 3:
    Actions:         (position-entity-ndom     entity: de0
                                               resources: (dom ndom eg ht brow))
    Effects:         (one-entity-has-been-positioned)
```

Figure 29: Method for OPTIONAL-POSITION-ENTITY

Ignoring "alternative 1" for a moment, let's consider why the planner would select "alternative 2" or "alternative 3" in this method. The POSITION-ENTITY-DOM task will show the location of an object in the signing space using the signer's dominant hand, and the POSITION-ENTITY-NDOM task will show an object's location using the signer's non-dominant hand. The use of the "(one-entity-has-been-positioned)" predicate will cause the first object whose location is shown in the signing space to be positioned using the signer's non-dominant hand, and all the rest of the objects will be positioned using the signer's dominant hand. In the "car parked between the house and the cat" example, this would cause the signer to show the house's location using his non-dominant hand and to show the cat's location using the dominant hand.

There are various factors (including the arrangement of the objects in the signing space and whether the signer is planning on performing a “secondary object” classifier construction) that could influence a signer’s choice of which hand to use when setting up objects in space during classifier predicates (Eccarius and Brentari, in press). However, for our prototype classifier predicate generator, we have used a very simplistic heuristic: When showing a set of surrounding objects before the main classifier predicate, use the non-dominant hand to show the location of the first surrounding object, and use the dominant hand to show the locations of the other surrounding objects. While this heuristic produced acceptable results in our prototype system (measured in the evaluation study discussed in Chapter 9), a future broad-coverage implementation of a classifier predicate generator should implement a more sophisticated decision criterion.

Now that we have considered why the system might choose between “alternative 2” or “alternative 3,” let’s consider why the system might choose “alternative 1.” In this alternative, the system does not show the location of the object – since the Actions field is <empty>, it would do nothing. The SHOP2 planner allows us to specify a “cost” for every primitive action (the default is a unit cost for each action), and the planner searches for a “cheapest plan” that successfully decomposes the initial set of tasks it has been given. Since “alternative 1” specifies no actions to be performed, it would be cheaper than “alternative 2” or “alternative 3.” We would like our generator to produce an ASL performance that is as short as possible yet which obeys the grammatical requirements of ASL and successfully decomposes all of the tasks passed to the planner. Although the planner will receive an OPTIONAL-POSITION-ENTITY task for every single Discourse Entity in the system’s Discourse Model, the default behavior (“alternative 1”) is to do nothing. If nothing in the decomposition of the CP-GENERATE task requires that the location of a particular Discourse Entity must be shown, then the system will create a plan that uses “alternative 1” in the method for OPTIONAL-POSITION-ENTITY.

However, if the classifier predicate that we wish to generate is trying to show a movement of an object relative to the location of some other objects, then all of those “relative-to” objects will need to be shown. For instance, in the “car parked between the house and the cat” example, we need to show the location of the house and the cat since the car is parking in a location that has been defined in terms of the house and cat locations. Whenever we show the location of an object in the signing space (using the POSITION-ENTITY-DOM or POSITION-ENTITY-NDOM) tasks, we cause the “positioned” and “identified” bits for that object’s Discourse Model to become true. So, one way to force the system to use “alternative 2” or “alternative 3” for some object is for a later classifier predicate in the schedule to include a precondition that the “identified” and “positioned” predicates are true for that object. In the next section, we will trace the execution of the planner for an example sentence, and the example will include a classifier predicate that uses the “positioned” and “identified” predicates in this way.

Example Trace of the Planner Execution

This chapter began by showing some of the early processing steps for the example sentence: “The car parked next to the house.” We’ve seen how information from this sentence is loaded from the lookup table and how the depicting canvas is produced. Based on this information, we can construct a set of Boolean predicates and initial high-level task list to be sent to the planner. As we can see in Figure 30, the system assigns symbolic names to each of the entities in the Discourse Model: “de000000” represents the “car” Discourse Entity and “de000001” represents the “house” Discourse Entity. The “isVehicle” and “isBulky” predicates indicate that the Discourse Entity bits with corresponding names are true. The OPTIONAL-POSITION-ENTITY and CP-GENERATE tasks have been described earlier in this chapter.

State information for the planner:

(isVehicle de000000)

(isBulky de000001)

(hasGhost de000000)

(hasGhost de000001)

Initial list of tasks to be completed:

(OPTIONAL-POSITION-ENTITY de000001)

(CP-GENERATE “park-23” de000000 “next to” (de000001))

Figure 30: Information Sent to Planner

Let’s first consider how the system would decompose the “(OPTIONAL-POSITION-ENTITY de000001)” task. The method shown in Figure 29 has three alternatives, and since the “(one-entity-has-been-positioned)” predicate has not been set, only alternatives 1 and 3 are possible. While alternative 1 is clearly cheaper, some later developments in the planning process will actually cause us to use alternative 3. So, we will show the decomposition using that alternative here. Figure 31 shows the decomposition of POSITION-ENTITY-NDOM, which is used in alternative 3.

```
POSITION-ENTITY-NDOM

Parameters:           Discourse Entity:    de0
                      Resource List:       resource_list0

alternative 1:
    Preconditions:    (isBulky de0)
    Actions:          (cp-locate-bulky-object-ndom       agent: de0
                                                         resources: resource_list0)

alternative 2:
    Preconditions:    (isAnimal de0)
    Actions:          (cp-locate-stationary-animal-ndom  agent: de0
                                                         resources: resource_list0)

alternative 3:
    Preconditions:    (isUprightFigure de0)
    Actions:          (cp-locate-upright-figure-ndom     agent: de0
                                                         resources: resource_list0)

alternative 4:
    Preconditions:    (isVehicle de0)
    Actions:          (cp-locate-vehicle-ndom     agent: de0
                                                  resources: resource_list0)

alternative 5: …
```

Figure 31: Code Sample from Method for POSITION-ENTITY-NDOM

Since the house is a bulky object, we will use alternative 1 in the decomposition of POSITION-ENTITY-NDOM, which includes the CP-LOCATE-BULKY-OBJECT-NDOM task. This task specifies that the signer should perform a classifier predicate in which they show the location of a bulky object using their non-dominant hand. The method which shows the decomposition of this task is shown in Figure 32.

```
CP-LOCATE-BULKY-OBJECT-NDOM
Parameters:          Discourse Entity:   de0
                     Resource List:      resource_list0

alternative 1:
    Preconditions:   (topicalized de0)
    Actions:         (locate-bulky-object-ndom        agent: de0
                                                      resources: resource_list0)

alternative 2:
    Actions:         (topicalization-phrase           entity: de0
                                                      resources: resource_list0)
                     (locate-bulky-object-ndom        agent: de0
                                                      resources: resource_list0)
```

Figure 32: Method for CP-LOCATE-BULKY-OBJECT-NDOM

This method checks whether the discourse entity has already been topicalized, and if not, then it performs a TOPICALIZATION-PHRASE task before performing the LOCATION-BULKY-OBJECT task. Since the "house" was not already topicalized at the start of planning, we must perform TOPICALIZATION-PHRASE. See Figure 33.

```
TOPICALIZATION-PHRASE
Parameters:          Discourse Entity:   de0
                     Resource List:      resource_list0

Preconditions:       (hasEg resource_list0)
                     (hasBrow resource_list0)
Actions:             (referringExpression     entity: de0)
                     (setLocation             target: (location eg)
                                              function: track_location
                                              source: audience_location)
                     (setBoolean              target:     (isRaised brow)
                                              function: track_boolean
                                              source: true_boolean)
```

Figure 33: Method for TOPICALIZATION-PHRASE

The decomposition for TOPICALIZATION-PHRASE first checks that the resource list contains the eye-gaze (“hasEg”) and the brow (“hasBrow”), and then it contains three primitive actions: “referringExpression,” “setLocation,” and “setBoolean.” The “referringExpression” primitive action will become part of the final plan schedule that the planner will produce – it specifies that the signer should perform an ASL referring expression for whatever Discourse Entity it specifies. The “setLocation” primitive action sets the value for the eye gaze location; it aims the eye gaze at the audience. This “audience_location” constant contains the location of the imaginary camera in the animation system. If the signer aims her eye-gaze at this location, then when a final animation is produced, she will appear to be making eye-contact with whoever is viewing the animation. The “setBoolean” primitive action sets the value for the “isRaised” Boolean feature of the signer’s eye-brows. When this value is set to true, then the signer’s eye brows are raised – this is important to do when performing a referring expression meant to topicalize an entity under discussion.

The CP-LOCATE-BULKY-OBJECT-NDOM decomposition also contained the LOCATE-BULKY-OBJECT-NDOM task. A decomposition for this task is shown in Figure 34. This task specifies how the signer should move their non-dominant hand to show a location in the signing space at which a bulky object is located. This task decomposition is quite similar to the one given for LOCATE-STATIONARY-ANIMAL shown in Figure 10 on page 21; the difference is that this task: checks that it controls the non-dominant hand (“hasNdom”), checks that “de0” is bulky (“isBulky”), specifies values for the non-dominant hand (“ndom”), uses the “bulky_object_ori_alignment” orientation alignment (described on page 154), and uses the “loose_c_handshape.”

```
LOCATE-BULKY-OBJECT-NDOM

Parameters:      Discourse Entity:   de0
                 Resources List:     resource_list0

Preconditions:   (hasNdom resource_list0)
                 (hasEg resource_list0)
                 (isBulky de0)
                 (hasGhost de0)
                 (topicalized de0)
Actions:         (setLocation     target: (location ndom)
                                  function: downward_arc_to_location
                                  source: (location (ghost de0)))
                 (setOrientation  target: (orientation ndom)
                                  function: get_to_final_orientation
                                  source: (orientation (ghost de0))
                                  alignment: bulky_object_ori_alignment)
                 (setHandshape    target: (handshape ndom)
                                  function: get_to_final_handshape
                                  source: loose_c_handshape)
                 (setLocation     target: (location eg)
                                  function: get_to_final_location
                                  source: (location (ghost de0)))
Effects:         (topicalized de0)
                 (identified de0)
                 (positioned de0)
```

Figure 34: Method for LOCATE-BULKY-OBJECT-NDOM

It is important to note that the Effects field of this method contains the predicates "(identified de0)" and "(positioned de0)." Later in our description of the planning process, we will see that one of the methods inside of the decomposition of the initial task "(CP-GENERATE 'park-23' de0000000 'next_to' (de000001))" will require that the "de000001" discourse entity is both "identified" and "positioned." This is the reason why we had to use the non-empty "alternative 3" decomposition of the OPTIONAL-POSITION-ENTITY task.

Now that we have fully decomposed the initial task "(OPTIONAL-POSITION-ENTITY de000001)," we can turn our attention to the other initial task that we passed to the planner: "(CP-GENERATE 'park-23' de0000000 'next_to' (de000001))." The decomposition for the CP-GENERATE task is shown in Figure 28, and since our Verb Sense Identifier is "park-23," we will be using "alternative 1" of this decomposition. Therefore, we will decompose the CP-GENERATE task into the EXPRESS-SEMANTICS-PARK-23. Figure 35 shows the decomposition for this task.

EXPRESS-SEMANTICS-PARK-23

```
Parameters:          Verb Sense Identifier:        string0
                     Agent Discourse Entity:       de0
                     Adjunct Preposition:          string1
                     Adjunct Discourse Entities:   de_list0

alternative 1:
    Preconditions:   (isLocativePreposition string1)
                     (isVehicle de0)
    Actions:         (cp-vehicle-park-relative-objects   agent: de0
                                                         adjunctList: de_list0
                                                         resources: resource_list0)

alternative 2:
    Preconditions:   (isVehicle de0)
    Actions:         (cp-vehicle-park      agent: de0
                                           resources: resource_list0)
```

Figure 35: Method for EXPRESS-SEMANTICS-BE-AT-LOC-34

If the preposition from the adjunct modifier phrase is a locative preposition, then we will use the CP-VEHICLE-PARK-RELATIVE-OBJECTS task, otherwise, we will use the CP-VEHICLE-PARK task. The string "next to" satisfies the "isLocativePreposition" predicate, and so we will use "alternative 1" in this decomposition. Just like the decomposition for CP-LOCATE-BULKY-OBJECT-NDOM, the task decomposition for CP-VEHICLE-PARK-RELATIVE-OBJECTS will check if its Discourse Entity has already been topicalized, and it will optionally trigger a TOPICALIZATION-PHRASE task. Since de000000 was not topicalized at the start of the planning process, a topicalization phrase will be triggered. CP-VEHICLE-PARK-RELATIVE-OBJECTS will also contain a task VEHICLE-PARK-RELATIVE-OBJECTS that specifies how the signer will perform a classifier predicate to show a vehicle parking at a location that has been defined in terms of some other objects in the signing space. See Figure 36.

```
VEHICLE-PARK-RELATIVE-OBJECTS

Parameters:        Discourse Entity:   de0
                   Relative Discourse Entities:   de_list0
                   Resources List:     resource_list0

Preconditions:     (hasDom resource_list0)
                   (hasNdom resource_list0)
                   (hasEg resource_list0)
                   (isVehicle de0)
                   (hasGhost de0)
                   ((topicalized de0) or ((identified de0) and (positioned de0)))
                   forall x in de_list0: (positioned x)
                   forall x in de_list0: (identified x)
Actions:           (vehicle-move-on-path     agent: de0
                                             resources: (subtract   resource_list0
                                                                    (ndom))
                   (final-platform-ndom      agent: de0
                                             resources: (ndom))
Effects:           (topicalized de0)
                   (identified de0)
                   (positioned de0)
```

Figure 36: Method for VEHICLE-PARK-RELATIVE-OBJECTS

This method has a precondition "((topicalized de0) or ((identified de0) and (positioned de0)))" – this means that the discourse entity "de0" must already be topicalized or it must be both associated with a location in the signing space and not have moved since the last time its location was indicated to the audience. If we want to perform a parking-vehicle classifier predicate, the audience needs to know what vehicle we are talking about. This can occur in one of two ways:

- If the car is already topicalized, then it is OK to begin the classifier predicate.
- If "the car" has already been assigned a location in space (and the car is still at this location), then if we begin to perform a classifier predicate shown a vehicle moving and parking, then the audience can figure out which entity we are referring to. (They can remember what object we last left in that location.)

The "de_list0" contains the list of other objects in the scene which we are parking relative to. (As far as the template knows, we could be parking near these objects, far from them, etc. – it doesn't matter. The details of where exactly the car is parking have already been determined by the scene visualization software, and this information should be inside the ghost locations for this object.) There is a precondition that "(identified x)" and "(positioned x)" is true for each item "x" in the "de_list0." What these preconditions enforce is that we have already shown the locations of the objects that we are parking relative to in space. This explains why we needed to use the non-empty decomposition of the OPTIONAL-POSITION-ENTITY task for the "house" Discourse Entity.

This method does not directly set the values of any articulators. Instead, it decomposes the VEHICLE-PARK-RELATIVE-OBJECTS task into a set of sub-tasks.

The method indicates that these sub-tasks should be performed in parallel because of the way in which it divides the resources among them. VEHICLE-MOVE-ON-PATH is given all of the resources in “resource_list0” except “ndom,” and FINAL-PLATFORM-NDOM is given a resource list that contains only “ndom.” The VEHICLE-PARK-RELATIVE-OBJECTS method is thus *partitioning* the responsibility for different articulators to each of these sub-tasks.

Figure 37 contains the decomposition for VEHICLE-MOVE-ON-PATH.

VEHICLE-MOVE-ON-PATH

```
Parameters:       Discourse Entity:   de0
                  Resources List:     resource_list0

Preconditions:    (hasDom resource_list0)
                  (hasEg resource_list0)
                  (isVehicle de0)
                  (hasGhost de0)
                  (topicalized de0)
Actions:          (setLocation      target: (location dom)
                                    function: track_location
                                    source: (location (ghost de0)))
                  (setOrientation   target: (orientation dom)
                                    function: track_orientation
                                    source: (orientation (ghost de0))
                                    alignment: top_thumb_front_knuckles)
                  (setHandshape     target: (handshape dom)
                                    function: track_handshape
                                    source: number_3_handshape)
                  (setLocation      target: (location eg)
                                    function: track_location
                                    source: (location (ghost de0)))
Effects:          (topicalized de0)
                  (identified de0)
                  (positioned de0)
```

Figure 37: Method for VEHICLE-MOVE-ON-PATH

The VEHICLE-MOVE-ON-PATH task specifies that the signer should use the dominant hand in a "Number 3" handshape to trace a motion path that follows the motion path of the Ghost for this vehicle. Along the way, the signer's eye-gaze will follow the motion path of the Ghost. This is the first planning operator shown that uses the "track_location" and "track_orientation" spatial transformation functions to specify values for the signer's hand. In this case, a movement is produced in which the hand traces a motion path for an object in the signing space.

Figure 38 contains the decomposition for FINAL-PLATFORM-NDOM. The FINAL-PLATFORM-NDOM task specifies that the signer moves the non-dominant hand (in a "Flat B" handshape) to a location just below the final location of the vehicle (the "upward_arc_to_below_location" spatial transformation function also specifies that the hand's motion path should approach this location from below with an upward arc-like movement). The orientation for the hand is specified using the "get_to_final_orientation" transformation function; in this case, the "upright_facing_forward" constant is used to set the goal orientation for the hand. Since the "top_insidepalm_front_knuckles" orientation alignment is specified for the hand, the result is that the hand will get to an orientation in which the palm is upright and the fingers are pointing forward.

VEHICLE-MOVE-ON-PATH and FINAL-PLATFORM-NDOM produce a performance in which the signer's dominant hand (in the "Number 3" handshape) will appear to "park" on top of the platform produced by the non-dominant hand.

```
FINAL-PLATFORM-NDOM

Parameters:       Discourse Entity:   de0
                  Resources List:     resource_list0

Preconditions:    (hasNdom resource_list0)
                  (hasGhost de0)
Actions:          (setLocation      target: (location ndom)
                                    function: upward_arc_to_below_location
                                    source: (location (ghost de0)))
                  (setOrientation   target: (orientation ndom)
                                    function: get_to_final_orientation
                                    source: upright_facing_forward
                                    alignment: top_insidepalm_front_knuckles)
                  (setHandshape     target: (handshape ndom)
                                    function: track_handshape
                                    source: flat_b_handshape)
```

Figure 38: Method for FINAL-PLATFORM-NDOM

As a result of the planning process described in this section, the following plan schedule is produced:

1. These three events will happen simultaneously:
    - A referring expression for the "house" will be performed.
    - The signer's eye-gaze will aim at the audience.
    - The signer's eye-brows will be raised.
2. These four events will happen simultaneously:
    - The non-dominant hand location shows the "house" location.
    - The non-dominant hand orientation shows the "house" orientation.
    - The non-dominant hand is changed to a "Spread C" handshape.
    - The signer's eye-gaze aims at the "house" location.
3. These three events will happen simultaneously:
    - A referring expression for the "car" will be performed.

- The signer's eye-gaze will aim at the audience.
- The signer's eye-brows will be raised.

4. These seven events will happen simultaneously:
    - The dominant hand shows the "car" motion path.
    - The dominant hand orientation shows the "car" orientation over time.
    - The dominant hand maintains a "Number 3" handshape.
    - The signer's eye-gaze follows the "car" motion path.
    - The non-dominant hand location moves to just below the "car" final location with an upward movement arc.
    - The non-dominant hand gets to an orientation where the palm is upright and the fingers point forward.
    - The non-dominant hand gets to a "Flat B" handshape.

Processing the Result from the Planner

The output of the planner is a schedule that is composed entirely of primitive actions: referringExpression, setLocation, setOrientation, setHandshape, and setBoolean. This schedule indicates groups of primitive actions that should occur in parallel and groups that occur in sequence. The non-planning portion of the classifier predicate generator uses this planner output to build a Partition/Constitute tree that specifies an ASL performance. There are some important steps to this tree-building process:

- The planner schedule is converted into a binary tree structure that contains Partition nodes to specify portions of the schedule that occur in parallel and Constitute nodes to specify portions that occur in sequence.

- All of the symbolic identifiers for discourse entities – e.g. de000000 – are converted into pointers to the proper Discourse Entity objects in the system's memory, and identifiers for ghost objects – e.g. (ghost de000000) – are converted into pointers to Ghost objects.
- Symbolic identifiers for locations or orientations of ghost objects – e.g. (location (ghost de000000)) – are converted into pointers to the appropriate Dynamic Location object or Dynamic Orientation object in the system's memory.
- Symbolic identifiers for any constants (e.g. the names of handshapes, the names of particular orientation alignments, etc.) are converted into pointers to the appropriate value in the system's memory. The values of these constants are recorded in libraries maintained by the non-planning portion of the generator.
- Occurrences of the "referringExpression" primitive action are converted into special "referring expression" leaf nodes in the P/C tree that is constructed.
- Occurrences of the other primitive actions are converted into leaf nodes in the P/C tree that specify the values of Articulators of the signer's body. The leaf node specifies their value in a "symbolic" manner; i.e. the name of the spatial transformation function is recorded in the leaf node, but the function is not yet "calculated" at this time.
    - The symbolic name of the spatial transformation function is recorded.
    - A pointer to the "source" parameter for the function is recorded.

    - Any other input parameters to the spatial transformation function (e.g. orientation alignments) are recorded in the leaf node.

- Special "null" leaf nodes are added to the P/C tree to represent portions of the schedule during which the value of some articulators are not specified.

Now that we have a P/C tree representation of the ASL animation performance, we can perform some necessary generation tasks. For example, whenever we encounter a "referringExpression" node in the tree, the system can use a referring expression generator to determine what type of output to produce. For our prototype generator, we simply produce a single ASL noun sign (e.g. HOUSE, CAR, etc.) for each discourse entity for which we are asked to select a referring expression. In a future broad-coverage implementation of a classifier predicate generator, a more sophisticated referring expression generator could be developed.

Handling "Null" Leaf Nodes

At this stage, we can also perform some post-processing on the "null" nodes in the tree. Whenever the value for the "isRaised" feature of the signer's eye-brows is not specified, we consider it to have a default value of false. Therefore, we replace "null" leaf nodes for the "isRaised" feature into leaf nodes that specify the value as being "false." Later in this chapter, we will discuss how the system selects what to do with the signer's eye-gaze when it is not directly specified in the P/C tree.

When we encounter a null leaf node for the location, orientation, or handshape of the signer's hand, we can often just allow the hand to remain in whatever position it was the last time it was specified. This approach works well in this prototype system since we are only producing short animations. If we were producing a longer ASL performance, it would be unnatural for the signer's hand to remain in a final position of a previous sign for too long a period of time. Eventually, the hand would return to a resting position.

By leaving the signer's hand in its last-specified position, the system occasionally produces animations in which the signer appears to be performing "secondary object" classifier predicate constructions (page 23). The signer holds the location of the "surrounding object" using her non-dominant hand while she performs a classifier predicate with her dominant hand. In this prototype system, this "holding behavior" does not occur for a principled linguistic reason; it is caused whenever either of the following two situations occurs:

- There is no referring expression between a classifier predicate using the non-dominant hand and a classifier predicate using the dominant hand.
- There is a referring expression between two such classifier predicates, but the referring expression only contains one-handed signs.

In future implementations of this classifier predicate generator, mechanisms should be explored for deciding (on linguistic grounds) when the signer should hold a classifier predicate produced with one of their hands while performing a classifier predicate with the other.[48]

Occasionally, leaving the signer's hand in the last-specified position produced awkward animations. Specifically, after the performance of a referring expression that

[48] Robert G. Lee (personal communication, May 16, 2006) noted the importance of these "secondary object" constructions during ASL classifier predicates and suggested their inclusion in this generator.

used both of the signer's hands, if the signer performed a one-handed classifier predicate, the other hand often got in the way or looked confusing. So, an additional post-processing step was added: After a referring expression that includes a two-handed sign, any "null" leaf node for one of the signer's hand should be turned into a leaf node that brings the hand down to a resting position.

Motion Planning and Surface Realization

Up until this stage of the generation process, the system has constructed and modified the information about the ASL performance in a symbolic manner – using identifier strings (inside the planner) or pointers to objects in memory (inside the non-planning component). This has allowed the system to isolate the classifier predicate templates (encoded as "task decomposition methods" in the planner) from any 3D coordinates or specific timing values for the animation performance. In any future broad-coverage implementation of a classifier predicate generator, this design would enable linguistic developers (with ASL expertise) to focus on the classifier predicate templates without needing to perform any 3D animation programming.

Calculating Spatial Data and Minimum Framesize for Leaf Nodes

Now that we are ready to produce an actual ASL animation from the P/C tree representation, we can convert from a symbolic to a numeric representation of the ASL performance – one which contains specific 3D coordinates and actual timing values for the nodes in the P/C tree. The current P/C tree contains three kinds of nodes:

- Referring expression leaf nodes that indicate where we should insert an ASL referring expression into the tree.
- Articulator-setting leaf nodes that specify how to set values for some articulators in terms of a spatial transformation function and a pointer to a source input value.
- Null leaf nodes that indicate a portion of the performance which is unspecified.
- Partitioning nodes that specify when two sub-trees should occur in parallel.
- Constituting nodes that specify when two sub-trees should occur in series.

The new P/C tree that we want to produce contains a corresponding set of nodes, but it will contain more specific 3D coordinate information and timing data. Specifically, leaf nodes should now contain a "minimum framesize" that indicates how many animation frames are required to perform the animation specified by that node. Null nodes are given a "minimum framesize" of 0, and non-leaf nodes are not yet assigned a "minimum framesize." For each "referring expression" leaf, we load a pre-recorded animation of an ASL noun sign for that discourse entity from the system's library of ASL signs. This animation is stored as a set of animation frames, where each frame specifies the joint angles for the signer's arms and hands. The number of frames in the animation data should be used to set the "minimum framesize" of the leaf.

For each articulator-setting leaf, we calculate the value of that articulator using the appropriate spatial transformation function. We produce a time-indexed array of values for that articulator.[49] To calculate some spatial transformation functions, we must know the value that the articulator had at the beginning of this leaf node. To find this value, we need to traverse backward in the P/C tree to find the last value that was specified for the articulator. Based on the spatial transformation function and the number of values specified inside of the "source parameter" to the function, we use a default framesize (somewhere between 30 and 60 frames) to set the "minimum framesize" for the node.

Calculating Leading Frames for Leaf Nodes

Now that the 3D location and orientation coordinates for the articulators have been loaded into the tree, we can check how many "leading frames" are needed for each leaf node. Leading frames are the number of animation frames that we have to add to the beginning of a node to allow all of the articulators to get into their starting values before the animation performance specified by that node. Leaf nodes that require their articulators to have a certain value on the very first animation frame of the node need to have some leading frames added. Pre-recorded animation nodes require leading frames, and so do articulator-setting nodes that use "track_..." spatial transformation functions. To estimate how many leading frames are needed at the beginning these nodes, the system checks what value the articulator had on the animation frame that immediately preceded this node. It then estimates how long it would take to move or rotate the articulator to get it to the starting value for this node.

[49] The time-index for entries in this array begins at 0 for each node in the tree – individual nodes do not yet know what their actual animation frame numbers will be in the final ASL animation performance. Further, this array of values does not necessarily store a value for every single frame of the animation; instead, it stores a value for certain key frames in the animation. For instance, to specify a motion-path for a hand during a node that should be 45 frames long, we might only specify a location at frames 0, 15, 30, and 45. This will allow the later human-body animation component greater freedom in planning the animation.

Calculating the Final Framesize for the Tree

Now that we have calculated a "leading frames" and an initial "minimum framesize" value for each leaf node in the tree, we can begin to calculate the timing values for the constituting and partitioning nodes. The size of a constituting node is simply the sum of the "leading frames" and "minimum framesize" values of its children. The size of a partitioning node is the size of its longest child. When we encounter a partitioning node in which the children have different sizes, the size of the shorter child must be increased to match the size of its longer sibling. This lengthening process recursively "stretches out" all of the nodes in the shorter child's sub-tree.[50] Once framesizes have been assigned to all the nodes in the tree, then we are ready to being motion planning.

Motion-Planning for the Hand and Arm Animation

This section describes how the P/C tree containing 3D animation coordinates and timing values is used to produce an animation of the ASL signer's hands and arms through motion planning.[51] The output of this motion-planning process is a list of animation frames that completely specify the rotation angles of the joints of the signer's hands and arms. The hand is specified using 20 rotation angles for the finger joints, and

[50] If we encounter a constituting node during this process, then the amount of stretching is divided equally among its children. If we encounter a partitioning node, then all of the children will need to be stretched the full amount. If we encounter a leaf node, then the way we stretch it depends on the type of leaf it is. If it is a recorded animation, we add several extra frames to the end of the animation to hold the signer's hands in their final pose. If it is an articulator-setting leaf that uses "track_..." functions to set its values, then we actually slow down the performance to stretch out the node. If it does not use "track_..." functions, then we just add some extra frames to the end of the node (just like we did for recorded animation nodes).
[51] This section describes joint work with Liming Zhao, a Ph.D. student in Computer Science at the University of Pennsylvania working in the Center for Human Modeling and Simulation. The text in this section is adapted from information written by Liming Zhao (personal communication, June 1, 2006).

the arm is specified using 9 rotation angles: 2 for the clavicle joint, 3 for the shoulder joint, 1 for the elbow joint, 3 for the wrist joint.

The input to this process is the P/C tree, which encodes animation data in different ways – depending on the type of leaf node. Recorded-animation leaf nodes (for the ASL referring expressions) already specify the animation data for the hands and arms in the proper format; so, these nodes are easy to process. Articulator-setting leaf nodes specify the handshape using 20 rotation angles (which is also already in the proper output format of the motion-planning process). However, these nodes specify the arm by giving a location for the center of the signer's palm and an orientation value for the palm. This means that during motion-planning, the system must find a set of clavicle, shoulder, elbow, and wrist angles that get the hand to this desired location and palm orientation.

Some Articulator-setting leaf nodes only specify a final location, orientation, and handshape that the hand must reach by the end of the node. Other Articulator-setting leaf nodes (that used "track_..." spatial transformation functions to set the values of their Articulators) contain a list of "landmark" location, orientation, and handshape values for the hand at several key animation frames along some motion path for that node.

Given the P/C tree as input, the motion-planning process must produce a set of rotation angles for the hand and arm joints that obey the following constraints:

- The signer's hand must reach all of the requested location, orientation, and handshape values given inside of the P/C tree.
- In-between the given location, orientation, and handshape values, some smooth interpolation should be calculated for the values.

- The arm pose for each frame must be as natural as possible, and the animation between frames must be smooth.

To plan an animation for the signer's body that obeys these constraints, we need a novel inverse kinematics (IK) which automatically favors natural arm poses. Inverse kinematics is a process by which an animation system searches for a set of joint angles in order to obtain a desired pose. In this case, we need to find a setting of joint angles for the shoulder, elbow, and wrist to get the signer's hand into a desired location/orientation.

For each arm, given a goal location and orientation for the center of the palm, we can compute the location where we need to place the signer's wrist. We use the wrist as the end-effector in our inverse-kinematics computation, and we first select an elbow swivel angle (based on the distance the wrist should be from the shoulder). We next compute a set of possible shoulder and wrist rotation angles in order to align the signer's hand with the palm orientation specified inside the P/C tree. Studies have shown that there are shoulder strength curves that describe the naturalness of arm poses (Mital and Faard, 1990; Pandya and Micocci, 1989). If we disregard any elbow angle that forces us to use impossible wrist joint angles[52], we can select the best possible arm pose that is collision free and with most shoulder strength (Zhao et al., 2005). In this way, for any given hand location/orientation, we compute the arm pose that is collision-free and most natural.

If we are given a "path" for the hand that contains a list of key landmark locations/orientations for the hand over time, then we can use interpolation to insert some

[52] We relaxed some of the wrist joint limitations of the virtual human character to make it easier for the hand to reach a larger set of location/orientation targets in the signing space. While this occasionally produces animations in which the ASL signer appears to have unusually limber wrist joints, none of the native ASL signers who evaluated the system (see Chapter 9) commented on this aspect of the animation.

extra landmark values (if the given set of landmarks is too sparse). We can also use interpolation to decide a path for the hand during the "leading frames" of a leaf node when the hand is moving from the final position of one leaf node to the starting position of the next one. To interpolate hand location values, we use B-Spline interpolation, and to interpolate palm orientation or handshape values, we use SLERP (Shoemake, 1985). Once we have computed a full set of joint rotation angles for each landmark, we interpolate among the joint rotation values to produce the required frames of animation.

Motion-Planning for Eye-Gaze and Eye-Brow Raising

Studies have shown that animated characters appear more natural if their facial models produce consistent eye movements (Lee et al., 2005). Thus, natural eye gaze is critical to the realism and believability of an animated character. This section describes how the P/C tree containing 3D animation coordinates and timing values is used to produce an animation of the ASL signer's eye-gaze, head-tilt (only as needed to accommodate comfortable eye-gaze angles), and eye-brow raising.[53]

The facial model for our classifier predicate generator is implemented using the Greta facial animation engine (Pasquariello and Pelachaud, 2001). Our model controls the motion of the signer's eye-brows, which can be placed in a "raised" or "flat" position depending on the value of the "isRaised" feature of the "brow" articulator in the P/C tree. The eye motor control repertoire of the model contains the following three behaviors: fixation on a 3D location in space around the signer's body, smooth pursuit of a moving

[53] This section describes joint work with Erdan Gu, a Ph.D. student in Computer Science at the University of Pennsylvania working in the Center for Human Modeling and Simulation. The text in this section is adapted from information written by Erdan Gu (personal communication, June 1, 2006).

3D location, and eye-blinking. Gaze direction is computed from the location values specified inside the P/C tree, and the velocity and time duration of the movement are determined by the timing values inside the tree.

In humans, horizontal gaze shifts greater than 20 degrees or vertical shifts greater than 10 degrees produce combined head and eye movement (Bahill et al., 1975). Therefore our model can also change the signer's head tilt when necessary to accommodate these types of gaze shifts.

Adults normally blink approximately 6-10 times per minute, but when a person is attentive to an object in the environment, the blinking rate will decrease. Therefore, when our model is performing a "fixation" or "smooth pursuit" with the eye-gaze, the rate of eye blinking is decreased.

Directional eye-gaze cues are frequent in face-to-face conversational interactions (Cassell et al., 2004; Gu and Badler, 2006), and studies have shown use of eye-gaze for turn-taking and information-grounding in sign language conversations (Baker, 1977; Coates and Sutton-Spence, 2001; Van Herreweghe, 2002). Thus, our ASL signer looks at the audience when performing referring expressions that introduce Discourse Entities prior to classifier predicates, and the signer also looks at the audience at the end of each signing performance. Whenever the signer's eye-gaze is not otherwise specified (i.e. when there is a "null" node in the tree for the eye-gaze), the default behavior is to look at the audience.

Displaying the Animation

The animation information produced during the hand/arm motion planning is integrated with the output of the eyes/face motion-planner to produce a full specification of the virtual human animation performance. This animation data is performed by a animated human character within the Center for Human Modeling and Simulation's Virtual Human Testbed (Badler et al., 2005). Because the Greta facial animation system produces a female head with light skin tone, a female virtual human body was chosen with matching skin. The character was dressed in a blue shirt and pants that contrasted with its skin tone. To make the character appear more like a conversational partner, the "camera" inside the virtual environment was set at eye-level with the character and at an appropriate distance for ASL conversation.

Figure 39 through Figure 50 show a sequence of 12 frames from an animation produced by the system. This is a performance of three classifier predicates that show a man walking between a tent and a frog. From the starting position (Frame 1), the signer performs the ASL sign "TENT" while looking at the audience with her eyebrows raised (Frames 2 to 3). Then she moves her left hand (in the "Spread C" handshape) into a position in space where the tent is located (Frame 4). With her left hand remaining in that position, the signer performs the sign "FROG" while looking at the audience with her eyebrows raised (Frames 6 and 7). She moves her right hand (in a "Hooked V" handshape) into a location where the frog is positioned (Frame 8). The signer performs the ASL sign "man" while looking at the audience and raising her eyebrows (Frames 8 and 9). Then she uses her right hand (in the "Number 1" handshape) to show the motion path of the man while following the path with her eyes (Frames 10 to 12).

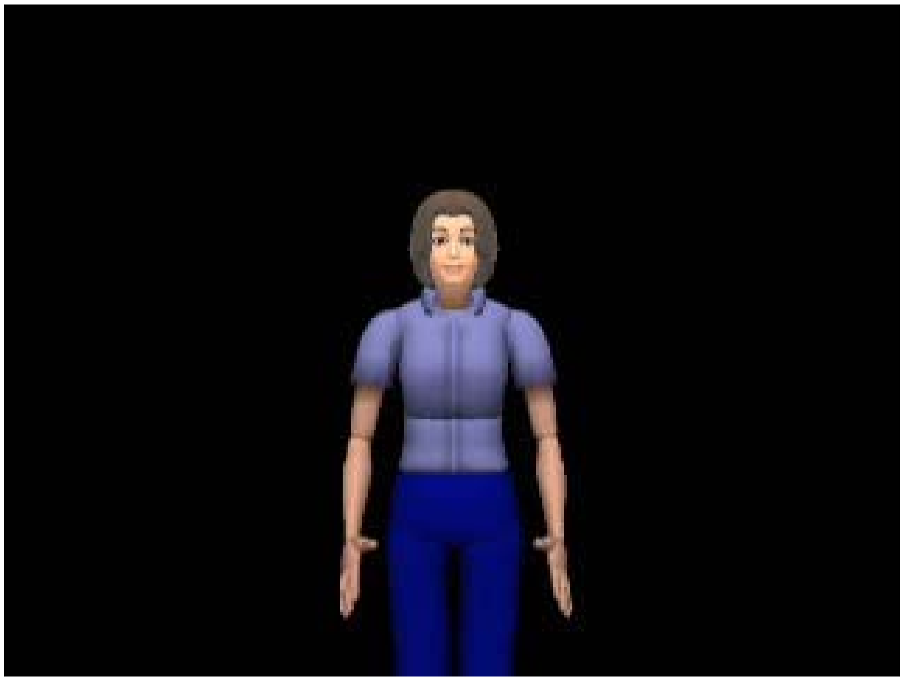


Figure 39: Output Animation Frame 1: Starting Position

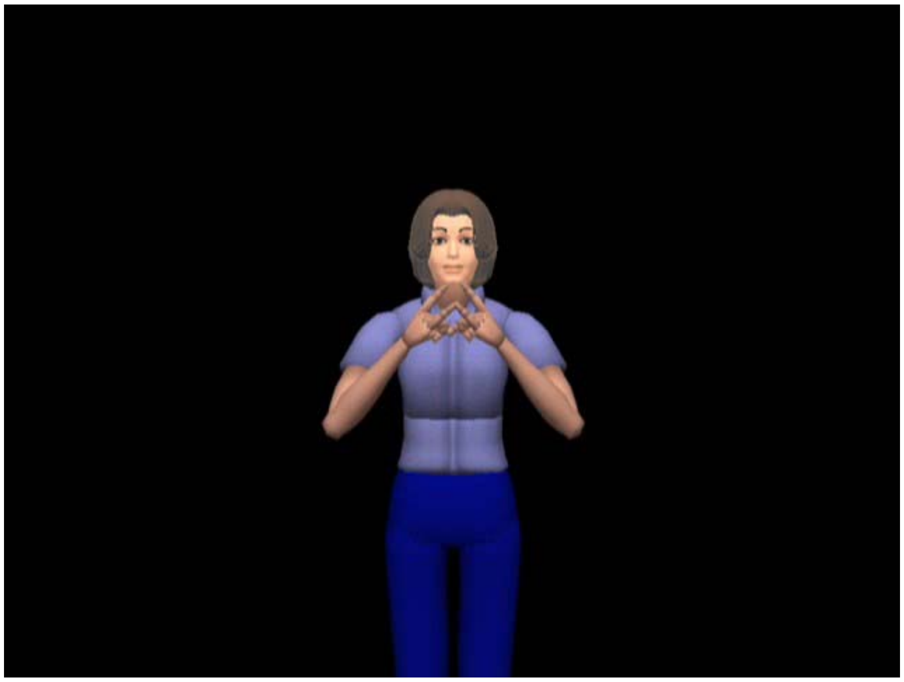


Figure 40: Output Animation Frame 2: ASL Sign TENT, part 1 of 2

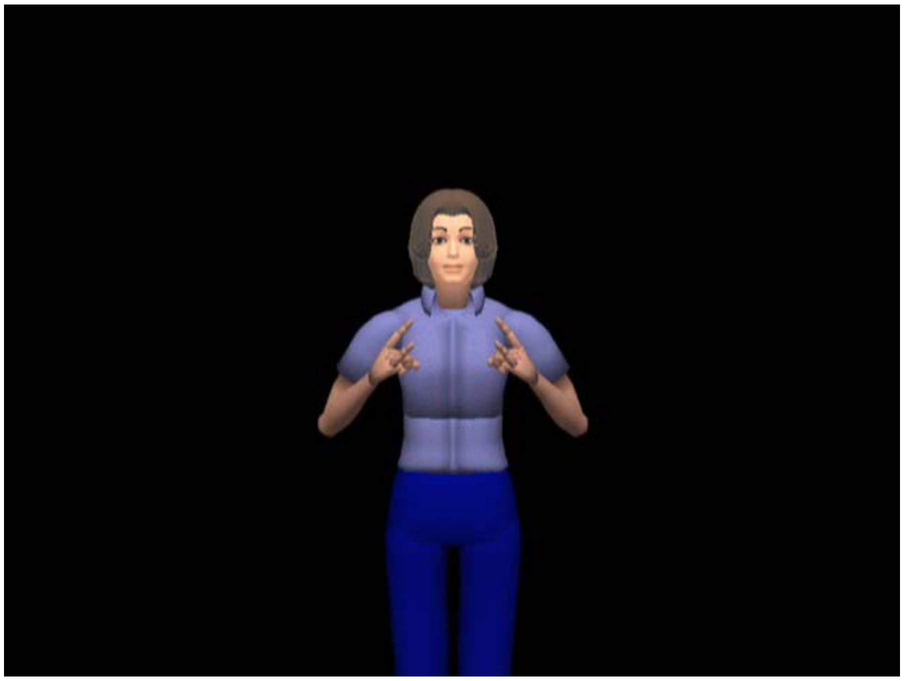


Figure 41: Output Animation Frame 3: ASL Sign TENT, part 2 of 2

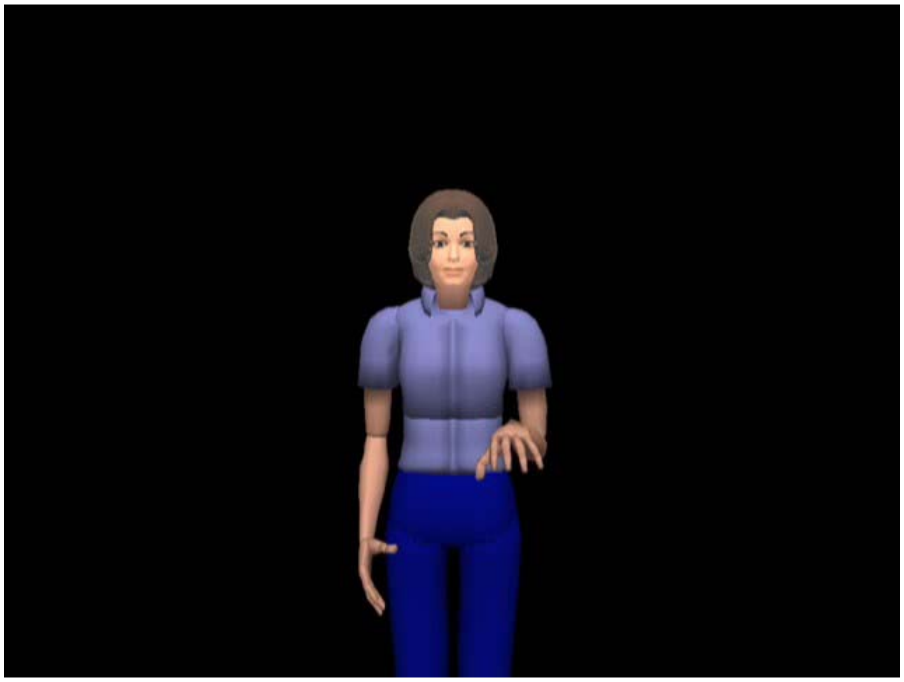


Figure 42: Output Animation Frame 4: Classifier Predicate for Tent Location

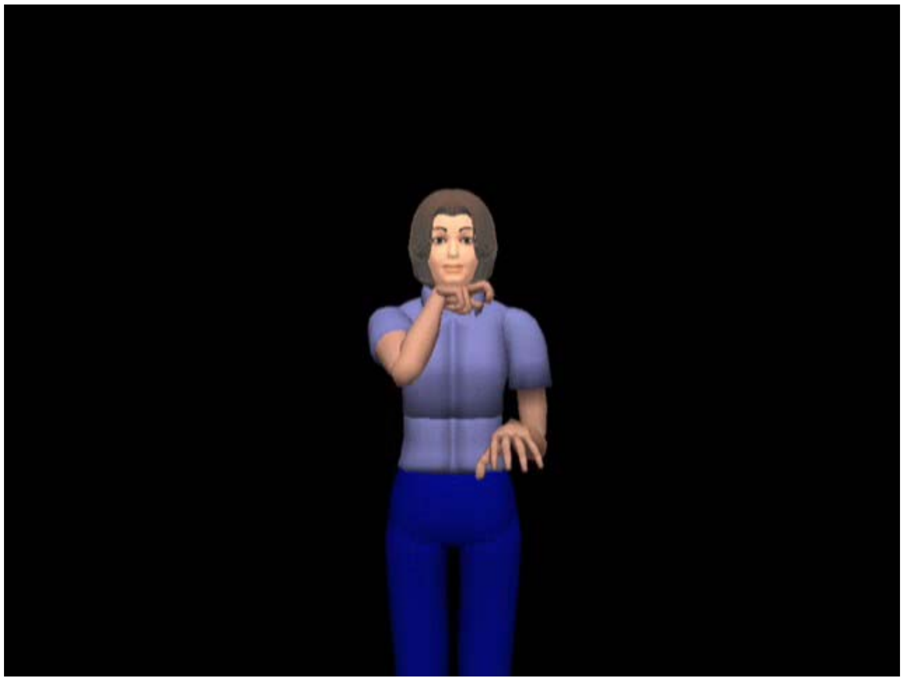


Figure 43: Output Animation Frame 5: ASL Sign FROG

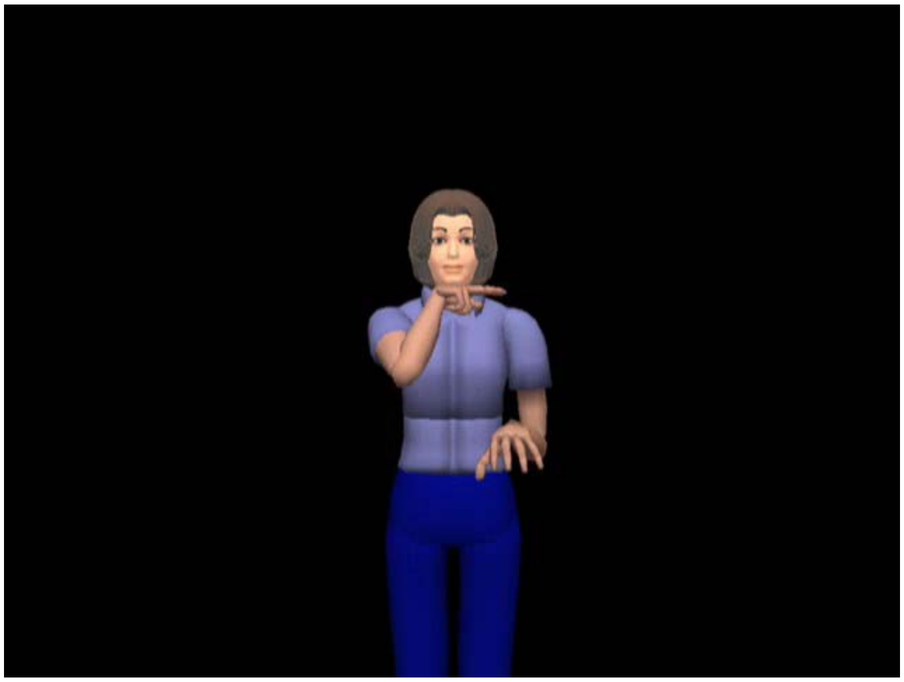


Figure 44: Output Animation Frame 6: ASL Sign FROG

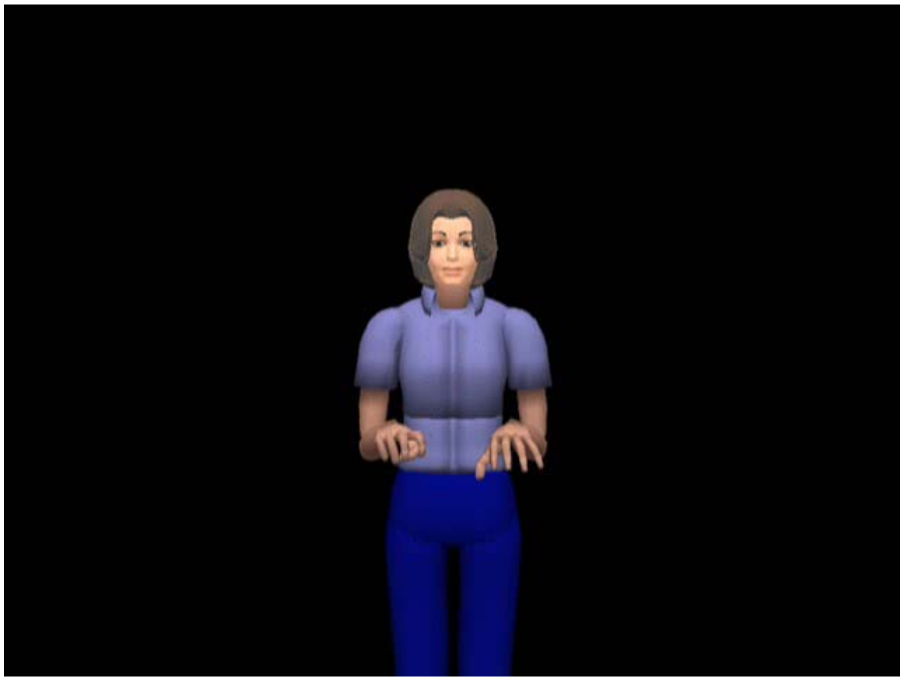


Figure 45: Output Animation Frame 7: Classifier Predicate for Frog Location

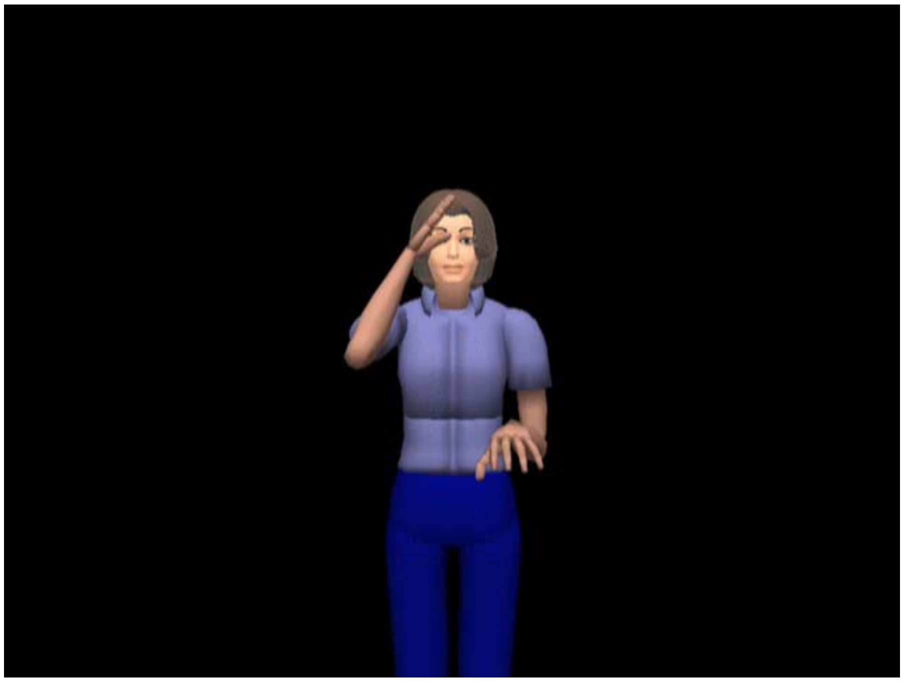


Figure 46: Output Animation Frame 8: ASL Sign MAN, part 1 of 2

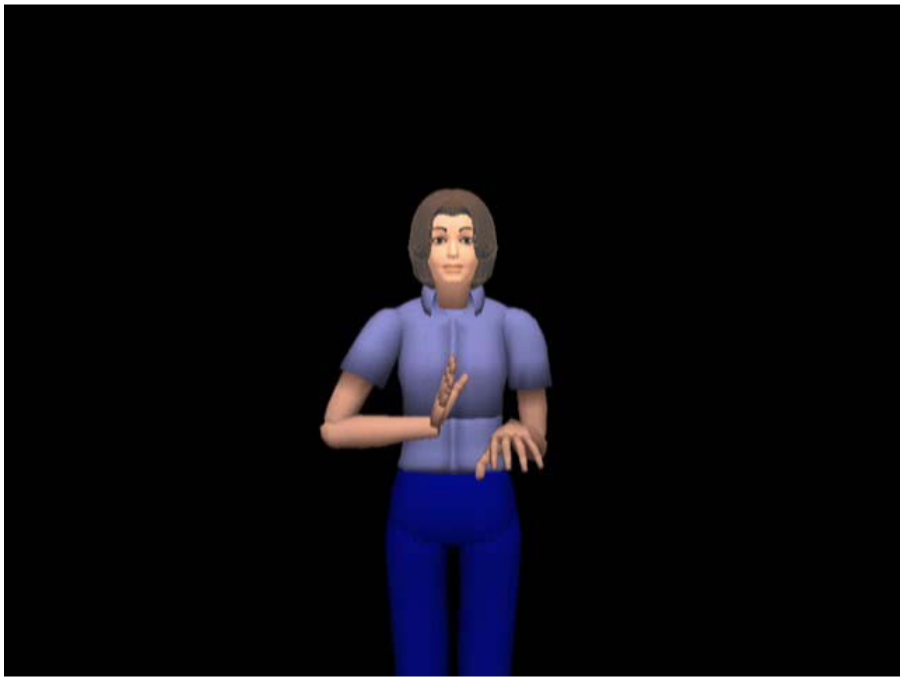


Figure 47: Output Animation Frame 9: ASL Sign MAN, part 2 of 2

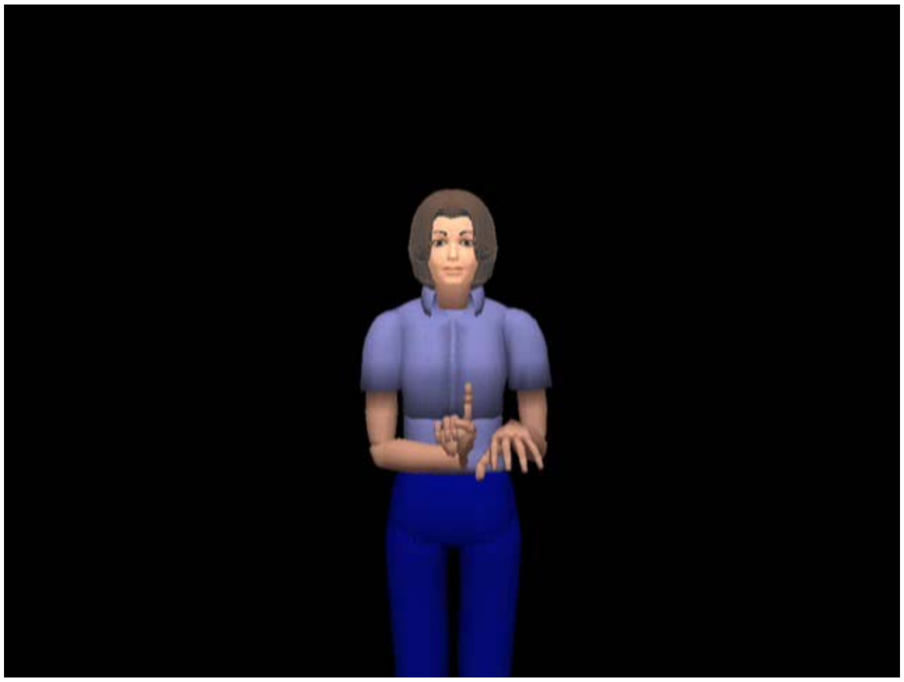


Figure 48: Output Animation Frame 10: Classifier Predicate for Man, part 1 of 3

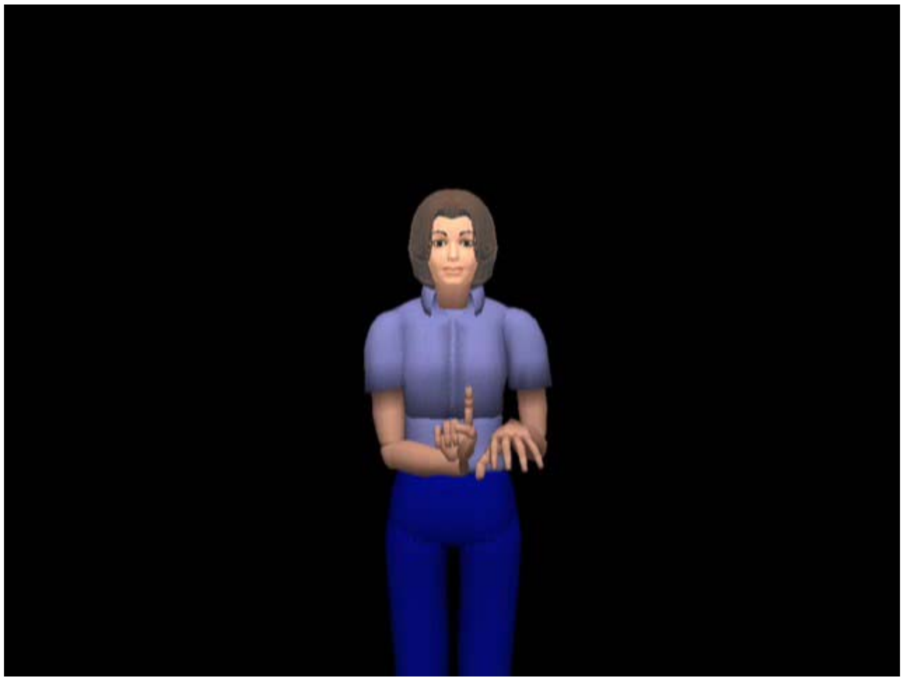


Figure 49: Output Animation Frame 11: Classifier Predicate for Man, part 2 of 3

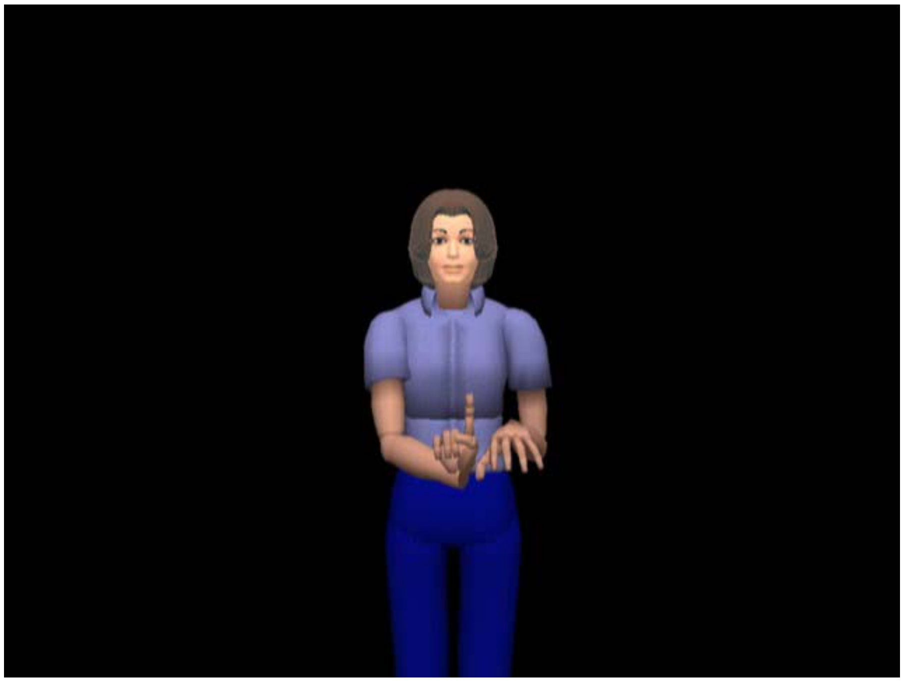


Figure 50: Output Animation Frame 12: Classifier Predicate for Man, part 3 of 3

Demonstrating Key Properties of the System

The implementation of a prototype classifier predicate generator has enabled us to demonstrate several of the "key properties" of the system and its design that were introduced in Chapter 1. The fact that a prototype classifier predicate generator was successfully implemented demonstrates that the system's design was ***well-specified*** and ***coherent***. The system produces animation output, but in order to confirm that the output is actually understandable and grammatical ASL classifier predicates, native ASL signers were asked to evaluate the system (see description of the evaluation study in Chapter 9) – thereby confirming whether the system is ***functional.***

Implementing this prototype classifier predicate generator has also allowed us to estimate the ***scalability*** of the design. After the non-linguistic framework of the system was implemented (which is a "fixed cost" for a broad-coverage implementation of the system), we tracked how much development effort went into the creation of the linguistic resources of the system (the list of classifier predicate templates and any spatial transformation functions they required in their definition). As we developed this "lexicon" of classifier predicate templates, we determined that it was possible to add new templates to the system without interfering with the functionality of the previously existing templates.[54] To expand the system's repertoire, additional "alternatives" were added to the method for CP-GENERATE (Figure 28) for any new verb word sense

[54] While this is always an important concern from a scalability perspective, it did not seem likely for this to be a problem in this system since each classifier predicate template is implemented as a planning operator. Planning algorithms are designed such that additions made to their list of operators should not cause the system to stop working – it merely gives the planner more ways to accomplish its goal. While the addition of planning operators may increase the processing time (since more possible plans may be considered), this is not unusual for a NLP system. Often the addition of linguistic resources (lexical entries, grammar rules, etc.) to an NLP system will increase the processing time of the system.

identifiers. Additional methods were added for any new tasks that appear in the decompositions in these new alternatives. (For instance, we may need to add additional "EXPRESS-SEMANTICS-…" methods to the system.) If any new types of classifier predicates had to be performed (e.g. if the system now needed to perform a classifier predicate to show the movement path of an upright-human-figure), then additional methods for this classifier predicate would be added to the system as well.

For the evaluation study (described in the next chapter), the prototype system needed to generate ten classifier predicate animations. As we extended the system to generate each additional animation, we tracked how many classifier templates had to be added to the system. Table 5 on page 223 in the next chapter contains a list of the ten animations; each animation was assigned a number (in the "#" column of that table). In Table 3 below, we indicate which of the classifier predicate templates in the system were used for each animation. (We have omitted any "EXPRESS-SEMANTICS-…" methods that were added during development.)

Table 3: Use of Classifier Templates in Each Animation

Classifier Predicate Template	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LOCATE-BULKY-OBJECT-NDOM	X				X	X		X		
LOCATE-ANIMAL	X					X				
VEHICLE-PARK-RELATIVE-OBJECT	X									
VEHICLE-MOVE-ON-PATH	X		X					X		
FINAL-PLATFORM-NDOM	X									
LOCATE-UPRIGHT-FIGURE-NDOM		X					X		X	
UPRIGHT-FIGURE-MOVE-ON-PATH		X				X	X		X	
LOCATE-FLAT-OBJECT-NDOM				X						X
LOCATE-STATIONARY-OBJECT				X	X					
LOCATE-UPRIGHT-FIGURE										X

Table 3 indicates how much we were able to re-use the classifier predicate templates – by the time we had finished extending the generator to produce the fourth animation, we had already created 9 of the 10 templates needed for the whole set.

In a similar way, we found that a fairly small set of spatial transformation functions allowed us to create templates for a diverse set of classifier predicates. For instance, the "downward_arc_to_location" function was used in the definition of several different classifier predicates that indicated the 3D location of entities in the signing space. This suggests that as the list of templates continues to grow (in a future broad-coverage implementation of a classifier predicate generator), then the amount of new *animation* development work would eventually "level off." Thus, the rate of growth of the set of spatial transformation functions with respect to the rate of growth of the set of classifier predicate templates is one metric of the scalability of the design. Table 4 shows how the spatial transformation functions in the system were used by each of the ten animations generated.

Table 4: Use of Spatial Transformation Functions by Each Animation

Spatial Transformation Function	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
rotate_rescale_recenter (location)	X	X	X	X	X	X	X	X	X	X
rotate_rescale_recenter (orientation)	X	X	X	X	X	X	X	X	X	X
track_location	X	X	X	X	X	X	X	X	X	X
Downward_arc_to_location	X	X		X	X	X	X	X	X	X
upward_arc_to_below_location	X									
get_to_final_location	X	X		X	X	X	X	X	X	X
track_orientation	X	X	X			X	X	X	X	
get_to_final_orientation	X	X		X	X	X	X	X	X	X
track_handshape	X	X	X			X	X	X	X	
get_to_final_handshape	X	X		X	X	X	X	X	X	X
track_boolean	X	X	X	X	X	X	X	X	X	X

On pages 151 to 156, we discussed the 11 spatial transformation functions that are listed in Table 4. Looking at the table, we can see that the first animation (for “the car parked between the house and the cat”) actually used all 11 of the spatial transformation functions. None of the nine animations that we generated subsequently required us to add any additional spatial transformation functions to the system.

Chapter 9:
Evaluation of a Prototype CP Generator

This chapter will discuss the evaluation of the prototype classifier predicate generator whose implementation was described in the previous chapter. In particular, this chapter will describe how a user-based evaluation study involving native ASL signers was designed and conducted.

Evaluating an ASL Generation System

The lack of an ASL writing system and the multichannel nature of ASL can make generation or MT systems which produce ASL animation output difficult to evaluate using traditional automatic techniques. Most automatic evaluation approaches compare a string produced by a system to some human-produced “gold-standard” string. While we could invent an artificial ASL writing system for the system to produce as output, it’s not clear that human ASL signers could accurately or consistently produce written forms of ASL sentences to serve as “gold standards” for such an evaluation.

Furthermore, real users of the system would never be shown artificial “written ASL”; they would see full animations instead. Thus, it was decided that a user-based evaluation study (in which subjects were asked to look at animation output) would be a more meaningful measure of the ASL animation system. For this reason, native ASL signers were asked to look at animation output produced by our prototype system and judge whether the animation was an understandable and grammatical ASL classifier predicate.

Questions to Ask

Signers were asked to rate each classifier predicate animation on ten-point scales for understandability, naturalness of movement, and ASL grammatical correctness. To obtain a more quantitative evaluation of how successfully the animation conveyed the information in the 3D scene being described, signers were also asked to complete a matching task after viewing each animation. After viewing a classifier predicate animation produced by the system, signers were shown three short animations showing the movement or location of the set of objects that were described by the classifier predicate. The movement of the objects in each animation was slightly different, and signers were asked to select which of the three animations depicted the scene that was described by the classifier predicate.

For future development purposes, signers were also asked to describe what aspects of the prototype classifier predicate generator's output should be modified to improve the overall quality of the ASL animation produced. The information obtained from this part of the study will be used to help direct future development efforts toward those portions of the system that native ASL signers felt require the most improvement.

Baselines for Comparison

If we had only asked subjects to give intelligibility/naturalness/correctness scores for animations produced by our system, then it's not clear how we could have interpreted the results. Since this prototype was the first generation system to produce animations of ASL classifier predicates, there are no other systems to compare it to in our study. The results of the evaluation are more meaningful if we are able to compare them to some baselines that represent expected upper- and lower-bounds on the system's performance.

Therefore, in addition to showing subjects animations produced by our classifier predicate generator, we also showed them animations created by:

- Asking a native ASL signer to perform classifier predicates while wearing a motion capture suit.
- Digitizing the coordinates of their hand, arm, and head movements during the performance.
- Using this information to create an animation of a virtual human character performing a classifier predicate.

The virtual human character used in the animations produced by the prototype classifier predicate generator is very similar in appearance to the virtual human character used in the animation produced by the motion capture. (The body of both is the same, but due to technical constraints, the head is slightly different in appearance.) Therefore, we can compare these two types of animation (while controlling for most variation in the visual appearance of the character).

To create a lower baseline for the evaluation study, we wanted to produce a set of animations that reflected the current state of the art in broad-coverage English-to-ASL translation. As we discussed in Chapter 3, there are currently no broad-coverage English-to-ASL MT systems; therefore, we decided to use Signed English animations as our lower baseline. We created animations of our virtual human character performing Signed English transliterations of the set of English sentences that our prototype classifier predicate system translated into classifier predicates.

Expectations

We expected the evaluation scores for the animations produced by our generator would be somewhat lower than the scores for the animations produced by the motion-capture process. We expected to outperform the Signed English animations; in particular, we expected the Signed English sentences to receive low ASL-grammaticality scores.

Definition of Success

While the two baselines discussed above make the results of this evaluation study more meaningful, they did not provide us a precise numerical definition of success. Since this was the first and only classifier predicate generation system evaluated against these baselines, it is not clear what level of performance constitutes a good or a bad performance. Obtaining a score relative to these baselines does make it easier for future researchers to compare the performance of their classifier predicate generators to the performance of this prototype, but the baseline-comparison is not particularly useful in determining whether the primary research claims of this project have been substantiated.

The key issue at stake during the evaluation study was whether the generator design is ***functional***. Thus, we were interested in whether ASL signers would agree that the animation output: (1) is actually a grammatically-correct and understandable classifier predicate and (2) is good at conveying the information about the movement of objects in the 3D scene being described. As this was a prototype-version of the system, we expected there to be animation details that could be improved in future work. However, since there are currently no NLP systems capable of generating ASL classifier predicate animations, any system that receives an answer of "yes" – to items (1) and (2) above – would be a significant improvement to the state of the art in English-to-ASL MT.

Conducting the Study

This section will describe how the user-based evaluation study of the prototype classifier predicate generator was conducted. To prepare for the study, it was important to select the set of sentences to evaluate, create the animations of the 3D objects described by each sentence, generate the classifier predicate animations, and create the Signed English animations (to serve as a lower-baseline). After obtaining Institutional Review Board clearance for the study, motion-capture data from a native ASL signer was collected and used to produce ASL animations. After preparing the evaluation survey materials (including a paper questionnaire and on-screen presentation of the animations), native ASL signers were recruited to evaluate the animations. During the study, the lab environment was controlled to minimize English influences on the ASL signers.

Selection of Sentences to Evaluate

Ten ASL classifier predicate animations (to be generated by the prototype system) were selected for inclusion in this evaluation study based on some desired criteria:

1. The animations should contain ASL sentences that consist of classifier predicates of movement and location (CPMLs) – not other types of classifier predicates.
2. The people and objects discussed in the sentence should require a variety of ASL classifier handshapes to be used: upright human figures (using the "Number 1" handshape), motorized vehicles (using the "Number 3" handshape), animals (using the "Hooked V" handshape), flat objects (using the "Flat B" handshape), bulky objects (using the "Spread C" handshape), and other stationary objects (using the "Thumb-Extended A" handshape – pictured in Figure 1 on page 21).
3. Some of the sentences should describe the static location of a set of people or objects, and some sentences should describe a movement event.
4. Some sentences should describe a single object, some should describe two objects, and some should describe three objects in a 3D scene.
5. Since the creation of a referring expression generator is not the focus of the prototype system, any referring expressions in the animations should simply be an ASL noun phrase consisting of a single sign. Some noun signs should be one-handed signs, and others should be two-handed signs.
6. A few of the sentences should contain a description of the same set of people or objects, but each sentence should discuss the objects moving in a different way.

The second, third, and fourth criteria promote the selection of a broad variety of classifier predicate animations for the study. The fifth criterion calls for a small group of the animations to discuss the same set of objects – allowing us to explore the performance of the system in more depth for a particular domain. Table 5 contains a list of the ten classifier predicate animations (along with the English sentence that served as the "input" to the generator for each). The first column of the table ("#") includes a number for each sentence – this identifier number indicates the order in which each sentence was added to the system, not the order that they are shown during the study.

The description for sentence #1 is quite detailed and explains the use of eye-gaze and brow raise throughout the performance. The remaining sentences are described more concisely. Although not stated explicitly, whenever an ASL sign is said to have been performed, the reader can assume that the eye-gaze was aimed at the audience and the brows were raised. The signer's eyes blink during the performance. During classifier predicates describing the static location of an object, the signer's eye-gaze jumps to the location of the object being described. During classifier predicates describing the movement path of an object, the signer's eye-gaze tracks the movement of the object.

Chapter 8 discussed heuristic rules (currently lacking a linguistically sophisticated implementation) used in this prototype. The signer looks at the audience at the end of each animation. After classifier predicates, the hand will remain in place (holding the location just shown) unless it is used to perform something else. After an ASL sign, any hand not used in the subsequent classifier predicate is lowered to a resting position.

Table 5: Sentences Included in Evaluation Study

#	English Sentence	Classifier Predicate Animation (description)
1	The car parks between the house and the cat.	1. Signer performs ASL sign "HOUSE" with both hands while looking at audience with raised brow. 2. Next, the non-dominant hand in "Spread C" handshape shows house location. (The hand remains until "CAR.") 3. Signer performs ASL sign "CAT" (with dominant hand only) while looking at audience with raised brow. 4. The dominant hand in a "Hooked V" handshape (and eye-gaze) shows the cat location; eye-gaze jumps to location. 5. The signer performs ASL sign "CAR" with both hands while looking at audience with raised brow. 6. At the same time, the dominant hand in "Number 3" handshape shows car's motion path (with eye-gaze following car), and the non-dominant hand in "Flat B" handshape creates platform for car to park on.
2	The man walks next to the woman.	1. ASL sign "WOMAN" performed with dominant hand. 2. Non-dominant hand in "Number 1" handshape shows the woman's location. (The hand remains here.) 3. ASL sign "MAN" performed with dominant hand. 4. Dominant hand in "Number 1" handshape shows man's path (moves forward straight past woman's location).
3	The car turns left.	1. ASL sign "CAR" is performed with both hands. 2. Dominant hand in "Number 3" handshape shows car's path (drives forward, turns left, and continues driving).
4	The lamp is on the table.	1. ASL sign "TABLE" is performed with both hands. 2. Non-dominant hand in "Flat B" handshape shows the table's location. (The hand remains here.) 3. ASL sign "LIGHT" performed with dominant hand. 4. Dominant hand in "Thumb-Extended A" handshape shows the lamp's location atop the table's location.
5	The tree is near the tent.	1. ASL sign "TENT" is performed with both hands. 2. Non-dominant ("Spread C") shows the tent's location. 3. ASL sign "TREE" is performed with both hands. 4. Dominant hand in "Thumb-Extended A" handshape shows the tree's location next to (near) the tent.

6	The man walks between the tent and the frog.	1. ASL sign “TENT” is performed with both hands. 2. Non-dominant hand in “Spread C” handshape shows the tent’s location. (The hand remains here.) 3. ASL sign “FROG” is performed with the dominant hand. 4. Dominant hand (“Hooked V”) shows frog’s location. 5. ASL sign “MAN” is performed with the dominant hand. 6. Dominant hand in “Number 1” handshape shows man’s path (moves forward straight between frog and tent).
7	The man walks away from the woman.	1. ASL sign “WOMAN” is performed with dominant hand. 2. Non-dominant hand in “Number 1” handshape shows the woman’s location. (The hand remains here.) 3. ASL sign “MAN” performed with dominant hand. 4. Dominant hand in “Number 1” handshape shows man’s path (moves diagonally forward away from woman).
8	The car drives up to the house.	1. ASL sign “HOUSE” is performed with both hands. 2. Non-dominant hand (“Spread C”) shows house location. 3. ASL sign “CAR” is performed with both hands. 4. Dominant hand in “Number 3” handshape shows the car’s path (drives up to a location next to the house and stops).
9	The man walks up to the woman.	1. ASL sign “WOMAN” is performed with dominant hand. 2. Non-dominant hand in “Number 1” handshape shows the woman’s location. (The hand remains here.) 3. ASL sign “MAN” performed with dominant hand. 4. Dominant hand in “Number 1” handshape shows man’s path (moves diagonally forward toward woman location).
10	The woman stands next to the table.	1. ASL sign “TABLE” is performed with both hands. 2. Non-dominant hand in “Flat B” handshape shows the table’s location. (The hand remains here.) 3. ASL sign “WOMAN” performed with dominant hand. 4. Dominant hand in “Number 1” handshape shows the woman’s location (standing next to the table).

Producing Three Visualizations for Each Sentence

For the “matching task” portion of the evaluation study, three animated visualizations were created for each sentence showing how the objects described in the sentence move in a 3D scene. One animation was an accurate visualization of the 3D location/movement of the objects, and the other two animations were “confusables” –

showing orientations/movements for the objects that would not match the ASL classifier predicate animations. The set of visualizations for each sentence is listed in Table 6, and the first visualization described for each sentence is meant to be the accurate one.

Table 6: Three Visualizations for Each Sentence

#	English Sentence	Three Visualizations (The first one is the correct one.)
1	The car parks between the house and the cat.	The house and the cat remain in the same location in each of these three visualizations: • A car drives on a curved rightward-forward path, and it parks at a location between a house and a cat. • A car drives between a house and cat, but it does not stop between them – instead, it continues driving off camera. • A car starts at a location between a house and a cat, but it drives to a location that is not between them.
2	The man walks next to the woman.	• Starting slightly behind and to the left of a woman, a man walks forward straight past her. She is facing forward. • A man walks diagonally forward and to the left away from a standing woman who is facing forward to the camera. • Starting slightly behind and to the left of a woman, a man walks forward straight past her. She is facing left.
3	The car turns left.	The car starts in the same location in each visualization: • A car drives forward, turns left, and continues driving. • A car drives forward and pivots left as it skids to a stop. • A car drives forward, turns left, continues driving, turns left, and continues driving. (The car performs a U-turn.)
4	The lamp is on the table.	The table remains in the same location in each visualization: • A table-top lamp sits atop a table. • A table-top lamp sits on the floor next to a table. • A hanging lamp is suspended above a table.
5	The tree is near the tent.	• A tree is next to a tent. (The tree is to the left of the tent.) • A tree is far in the background diagonally behind and to the left of a tent that is in the foreground. • A tent is far in the background diagonally behind and to the left of the tree that is in the foreground.

6	The man walks between the tent and the frog.	The frog and tent remain in the same location in each: • A man walks forward to the camera between a frog on the left and a tent on the right. (The man walks perpendicular to an imaginary line connecting the frog and the tent.) • A man walks diagonally forward-left starting from a position that is between the frog and the tent. • Starting at a location to the left of the frog (not between it and the tent), a man walks diagonally forward-right.
7	The man walks away from the woman.	• A man walks diagonally forward and to the left away from a standing woman who is facing forward to the camera. • Starting slightly behind and to the left of a woman, a man walks forward straight past her. She is facing forward. • Starting slightly behind and to the left of a woman, a man walks forward straight past her. She is facing left.
8	The car drives up to the house.	The house remains in the same location in each visualization: • A car drives left-to-right up to a house and stops next to it. • A car drives forward to camera and skids to a stop turning left at a location that is near the house. • A car drives forward straight past a house (without stopping) and continues driving off camera.
9	The man walks up to the woman.	• A man walks diagonally forward-right toward a woman (who is facing forward) and stops at a location near her. • A man and a woman walk toward each other and stop. • A man walks up to a woman who is facing him.
10	The woman stands next to the table.	The table remains in the same location in each visualization: • Facing forward, a woman stands to the left of the table. • A woman stands to the left of the table while facing it. • Facing forward, a woman stands in front a table.

Because we wanted our evaluation to primarily test the planning-based portion of the classifier predicate generator (and not the referring expressions created from ASL noun signs), the set of objects that appeared in all three visualizations was the same. In this way, it would be the movement and orientation information conveyed by the classifier predicate (and not the object identity conveyed by the referring expression) that would distinguish the correct visualization from the confusables. One exception was made in order to make one of the confusables for sentence #4 more plausible. One visualization needed to show a lamp floating above a table. A hanging lamp suspended from a ceiling was used in this visualization, instead of the table-top lamp used in the other two cases.

Creating the Classifier Predicate Animations

For each of the 10 sentences, "lookup table" entries were created for the classifier predicate generation system. English Predicate Argument Structures and Discourse Model settings were created for each sentence. (The appropriate bits were set to true to indicate which objects were vehicles, animals, etc., and the "topicalized," "identified," and "positioned" bits were set to false for all of the Discourse Entities.) To set the location and orientation values inside of the Visualization Scene for each sentence, the actual 3D data from the "correct" visualization animation for that sentence was used. The same Transformation Matrix was used for all of the sentences to position the animated visualization into the space in front of the signer. The necessary classifier predicate planning operators had been added to the system during the prototype implementation work described in Chapter 8. The system generated animations of ASL classifier predicates using the planning-based approach described in that chapter. The animations were saved as files that could be played back during the evaluation study.

Creating the Signed English Animations

To create the animation of a Signed English transliteration for each English sentence, some additional signs were added to the prototype generator's library of ASL signs. The sequence of signs in each Signed English transliteration was "strung together" as a right-branching binary Partition/Constitute tree that contained only Constituting nodes and recorded-animation leaf nodes. The tree contained one recorded-animation leaf node for each Signed English sign in the sentence.

This P/C tree was passed to the post-planning component of the generator to produce an animation output. The system inserted smooth and natural transitional movements for the arms and hands between each Signed English sign in the sentence. The string of glosses for the Signed English transliteration of each sentence is shown in Table 7.

Table 7: Signed English Transliterations

#	**English Sentence**	**String of Signed English Glosses**
1	The car parks between the house and the cat.	THE CAR PARK BETWEEN THE HOUSE AND THE CAT
2	The man walks next to the woman.	THE MAN WALK NEXT-TO THE WOMAN
3	The car turns left.	THE CAR TURN LEFT
4	The lamp is on the table.	THE LIGHT IS ON THE TABLE
5	The tree is next to the tent.	THE TREE IS NEAR THE TENT
6	The man walks between the tent and the frog.	THE MAN WALK BETWEEN THE TENT AND THE FROG
7	The man walks away from the woman.	THE MAN WALK FROM THE WOMAN
8	The car drives up to the house.	THE CAR DRIVE TO THE HOUSE
9	The man walks up to the woman.	THE MAN WALK TO THE WOMAN
10	The woman stands next to the table.	THE WOMAN STAND NEXT-TO THE TABLE

Collection of ASL Motion-Capture Data

The Center for Human Modeling and Simulation at the University of Pennsylvania includes a motion-capture room with a ReActor II system (Ascension Technologies, 2006) and a pair of wireless CyberGloves (Immersion Corp., 2006).[55] To record human

[55] The motion-capture data collection was joint work with Jan Allbeck and Liming Zhao, Ph.D. students in Computer Science at the University of Pennsylvania working at the Center for Human Modeling and

movements, 30 infrared-emitting markers are attached (with Velcro to a spandex suit) at key locations on the body, and sensors around the room triangulate the position of each marker at a rate of 30Hz. The gloves record 22 joint-angle measurements for each hand using resistive band-sensors and transmit their data to a host computer via Bluetooth. The gloves require calibration of a numerical offset and gain for each of the 22 joint values – a somewhat tedious process that must be repeated to some degree each time the gloves are put on. While the system captures head movements (via sensors placed on a headband worn by the human being recorded), the system does not capture eye-gaze or facial expressions.

Software plug-ins for both the ReActor and CyberGlove systems allowed data to be streamed from the devices into Autodesk MotionBuilder, a 3D character animation program. Within MotionBuilder, the motion-capture data is applied to a human model and a stream of joint angles for the human's skeleton is generated. This information is recorded in the form of a "BVH" (or "Biovision Hierarchical") file, a standard file-format used to record joint rotation information from motion-capture data.

The motion-capture room initially contained several curtains and pieces of equipment that broke-up the line of sight between where the researchers needed to be to operate the computers and where a research participant needed to stand to perform the ASL sentences. To make the room more accessible for a deaf research participant, the lighting was increased and various curtains and equipment moved to create a line of sight between all of the people who needed to be present to collect the motion-capture data.

Simulation. (Jan Allbeck is also the Associate Director of the Center.) The first two paragraphs of this section (describing the technical details of the motion-capture equipment and the recording process) are adapted from information written by Jan Allbeck (personal communication, May 30, 2006).

After obtaining Institutional Review Board approval for the study, a native ASL signer was recruited to perform ASL classifier predicates while wearing the motion-capture body suit and data gloves. To ensure that we were actually recording ASL, it was important that our contributor was a *native* signer – someone who learned ASL in early childhood through interaction with deaf family members or experiences at a residential school for the deaf.

As discussed by Neidle et al. (2000), when recording ASL data for research purposes, it is important to minimize the English influences in the environment to produce a setting in which the signer is not prone to code-switch to a more English-like form of signing. Since a native ASL-signing researcher was not available to interact with the contributor, a pair of interpreters facilitated communication. Interpreters were requested from the interpreter referral service with native or near-native ASL fluency, and they were informed before the session that the deaf contributor's preferred mode of communication was ASL (and not Signed English or one of the other more English-like signing communication systems).

The signer put on the motion-capture suit and gloves, and the gloves were calibrated. Before the data collection, the signer moved their arms and signed with the interpreters. The signer reported that the suit and gloves were comfortable and did not feel like they were impeding the signer's movement while signing.[56] The gloves are made of spandex with thin sensor strips on the back of the joints; the fingers had sufficient freedom of movement that the interpreters understood the signer's fingerspelling and signing.

[56] While the signer reported that the body suit and gloves did not impede signing, further study may be needed to see if wearing the motion-capture equipment influenced the signer's ASL performance.

To elicit the ASL classifier predicate sentences, the signer was shown the “correct” animated visualization for each of the ten sentences – showing a set of objects moving in a 3D scene. The signer was asked to describe the arrangement of the objects in the scene (their locations and orientation), but she was told that it was not important to describe the appearance (size, shape, color) of the objects in the scene. The signer was asked to use ASL to describe the scene – as she might describe it to another ASL signer that she were having a conversation with. In fact, asking a signer to imagine that they are conversing with another ASL signer (as a means of encouraging them not to code-switch to English-like signing) was a recommendation of (Neidle et al., 2000).

Creating Animations from the Motion-Capture Data

As this was the first project at the Center for Human Modeling and Simulation to use both the ReActor and the CyberGloves to record data simultaneously, there were several equipment and software challenges encountered – particularly with the CyberGloves. Finding the proper settings for the gain/offset values for the 22 joints on each glove was a difficult and slow calibration. When the data collection finally began, sensitive on/off switches and battery power limitations led the CyberGloves to turn off multiple times during the motion-capture session, forcing a several minute delay while the wireless connections were reestablished and the gloves reconnected to the motion-capture software. Due to challenges in collecting accurate handshape data, it was necessary to later go through the recorded BVH files and manually modify the signer’s handshapes.

Even when there are no technical challenges, data collected from a motion-capture session frequently requires manual editing of the animation after the data-collection session is over. If the placement of markers on the body suit is not perfect, if the actual human's body proportions differ from the MotionBuilder human model, or if there is incorrect location-triangulation when markers are occluded, then it can be difficult for the MotionBuilder software to calculate accurate human skeleton angles from the motion-capture data it receives. The effect is that some of the human's joint angles in the motion-capture animation can appear distorted, and finding a proper set of transformations to apply to the joint data during post-processing to compensate for this distortion is difficult. In fact, if it were not important for our project to base the animated character's movements on a human signer, it would have likely been much faster to manually animate a human figure to perform ASL sentences than it was to remove distortions and errors from the motion-capture data that was recorded.

After the data from the motion-capture was cleaned up and corrected, a virtual human character was used to perform the BVH file.[57] Although the clean-up process improved the motion-capture data, the resulting animation was still not a perfect reproduction of the human signer (and the animation still had occasional "jitter" in its movements). While the virtual human body, clothing, lighting, and camera angle used for the motion-capture data was identical to those used for the classifier predicate and Signed English animations, the signer's head was slightly different. While a Greta head was used for the classifier predicate and Signed English animations, software limitations prevented this head from being used for the motion-capture data. Therefore a similar-looking (though not identical) female head with light skin and dark hair was used instead.

[57] Using BVH data to animate the virtual human was joint work with Liming Zhao, a Ph.D. student in Computer Science at the U. of Pennsylvania working in the Center for Human Modeling and Simulation.

Survey Questionnaire and Computer User-Interface

Having created the 10 classifier predicate animations, 10 Signed English animations, 10 motion-capture animations, and 30 animated visualizations of 3D objects, we were ready to create the survey materials. To make it more convenient to show the animations to research subjects (and to make sure that the proper three visualization alternatives are shown for each animation), a simple on-screen user-interface was created that displayed one signing animation and three alternative object-visualizations on the screen at a time. See Figure 51 for a screenshot of one such "slide." After creating a slide for each of the 30 animations, the slides were placed in random order in a presentation. A user could re-play the signing animation and the visualizations as many times as desired before pressing the "Next" button at the bottom of the slide to go to the next signing animation.

Subjects recorded their responses by circling a choice on a paper survey form. The subject was asked to rate each of these 30 animations on a 1-to-10-point scale for ASL grammatical correctness, understandability, and naturalness of movement (i.e. whether the signer moves like a real person or like a robot). The subject was also asked to select which of the three animated visualizations (choice "A," "B," or "C" on the slide) matched the scene described in the virtual human character's signing performance.

At the end of the 30 slides, 3 additional slides were added that contained animations produced by the classifier predicate generator. These three slides only showed the "correct" animated visualization for that sentence. For these last three slides, subjects were asked to comment on the animation's speed, colors/lighting, hand visibility, correctness of hand movement, facial expression, and eye-gaze. Signers were also asked to write any comments they had about how the animation should be improved.

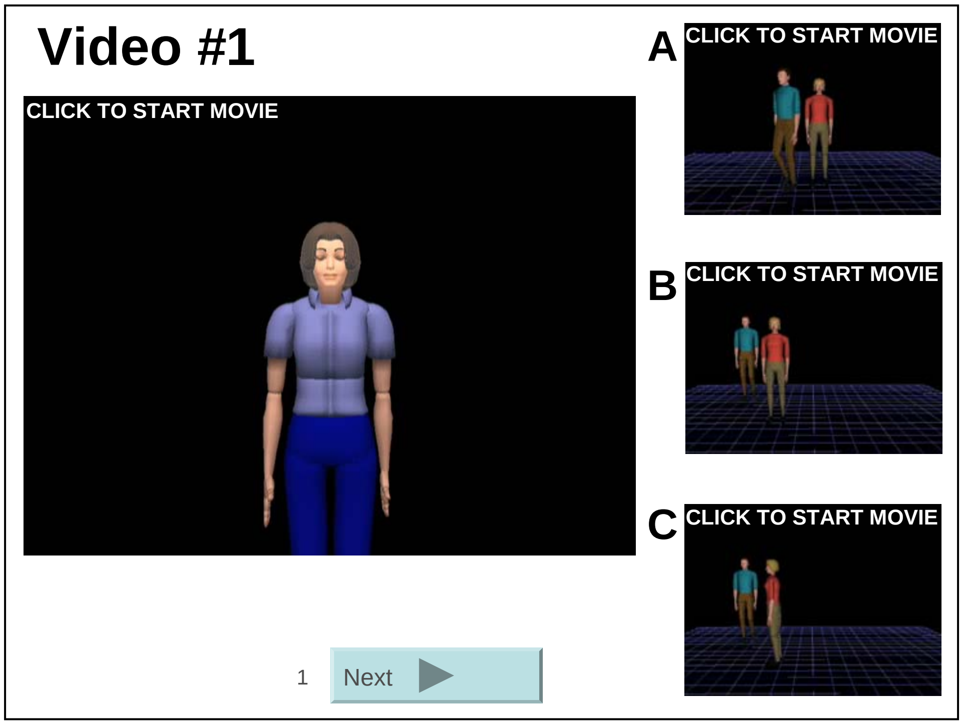


Figure 51: Slide Showing a Signing Animation with 3 Visualization Choices

Recruitment and Screening of Research Subjects

After obtaining Institutional Review Board approval, we recruited and screened subjects for the evaluation-survey portion of the study. Subjects were recruited through personal contacts in the deaf community who helped to identify friends, family, and associates who met the screening criteria. Participants in the study had to be native ASL signers. Subjects were preferred who had learned ASL since birth, had deaf parents that used ASL at home, and/or began attending a residential school for the deaf as a child (where they were immersed in an ASL-signing community). Table 8 summarizes the childhood ASL experiences of the fifteen subjects that participated in the study. The cover page of the questionnaire asked questions about their ASL experiences as children, educational experiences, and whether their parents or family were deaf or ASL signers.

As an informal check on participants' level of ASL fluency, a native ASL signer was present during 13 of the 15 sessions to converse in ASL with each participant.[58]

Table 8: Research Subjects' Childhood ASL Experiences

Subject	Learned ASL Since Birth?	Deaf Parents that Used ASL at Home?	Attended Residential School for the Deaf?
1	X	X	X
2	X	X	X
3	X	X	X
4	X	X	X
5	X	X	X
6	X	X	X
7	X	X	X
8	X	X	X
9	X	X	
10	X		X
11		X	
12			X
13			X
14			X
15			X

Avoiding English Influences in the Experimental Environment

As we discussed earlier in this chapter, when eliciting signing performances or when asking for grammaticality judgments from ASL signers, it is important to minimize characteristics of the experimental environment that could prompt the signer to produce or accept English-like signing. Neidle et al. (2000) discuss a variety of issues that should be considered when collecting linguistic data from ASL signers. Most of these issues have a common theme: that it is best to create an environment in which the research subject is surrounded by ASL signing and is not exposed to English text or non-native

[58] We noticed (from the written comments participants gave at the end of the study) that many of them had strong English skills. In future work, we may design studies in which subjects are also screened for their level of English literacy so that we can better represent the target users of English-to-ASL MT software.

English-like signing. During the evaluation study, instructions were given to participants in ASL, and a native ASL signer was present during 13 of the 15 sessions to answer questions or explain experimental procedures. This signer engaged the participants in conversation in ASL before the session to help produce an ASL signing environment.

While there was English text on the paper-based questionnaire, it was designed so that signers only needed to circle responses to an identical set of four questions for each of the 30 animations they evaluated during the study. The text on the questionnaire was not full English sentences, but instead consisted of short phrases serving as cues identifying each question. See Figure 52. The questionnaire was made repetitive by design – so that subjects would not continue to re-read the questions each time.

Participants were given instructions in ASL about how to respond to each of the survey items. For grammaticality, they were told that "perfect ASL grammar" would be a 10, but "mixed-up" or "English-like" grammar should be a 1. For understandability, they were told that sentences which were "easy to understand" should be a 10, but sentences that were "confusing" should be a 1. For naturalness, they were told that animations in which the signer moved "smoothly, like a real person" should be a 10, but animations in which the signer moved in a "choppy" manner "like a robot" should be a 1.

Video #1:	Good ASL grammar? (10=Perfect, 1=Bad):	10	9	8	7	6	5	4	3	2	1
	Easy to understand? (10=Clear, 1=Confusing):	10	9	8	7	6	5	4	3	2	1
	Natural? (10=Moves like person, 1=Like robot):	10	9	8	7	6	5	4	3	2	1
	Which picture/movie on the right matches?	A	B	C							

Figure 52: One of the 30 Identical Sets of Questions on the Evaluation Survey

Although the use of a user-controlled onscreen animation slideshow and a paper-based questionnaire led to the use of some English text on the questionnaire, this was considered better than the alternative: having a researcher sit with each participant during the experiment asking questions in ASL and taking notes based on the signer's responses. To avoid exposing the participant to English-like signing, the researcher would need to be a native ASL signer (which was not feasible for this study). Even if a native ASL signer was available to conduct the survey, there is a danger that the participants' interaction with the researcher during the evaluation would have affected their grammaticality judgments. By producing an overly clinical or official setting, code-switching to English-like signing could have been prompted in some participants (Neidle et al., 2000). Further, the participants may have been less willing to give negative opinions about the animations if they felt that someone involved in creating them was conducting the survey.

Results of the Evaluation Study

We will begin this section by clarifying some terminology to be used consistently throughout the rest of the chapter. There were three **groups** of animations tested: animations produced from motion-capture data (which we will abbreviate "MO" in this chapter), classifier predicates generated by the prototype system (abbreviated "CP"), and Signed English animations (abbreviated "SE"). There were ten **sentences** that were included in the study (sentences #1 to #10). Therefore, there were 30 **animations** evaluated in the study (3 groups × 10 sentences). To avoid confusion, we will use the term **visualizations** to refer to the animations of moving 3D objects that were created for each of the 10 sentences. Since there were 3 visualizations created for each sentence, there also happen to be 30 visualizations that were used throughout this study.

Since there were 15 **participants** in the study, and each evaluated all 30 animations, we therefore collected a total of 450 **responses**. (We also obtained personal information about each participant and written feedback about some CP animations, but for now we are only considering the numerical survey responses.) Each response consisted of a score in four different **categories**: grammaticality (1 to 10 score), understandability (1 to 10 score), naturalness of movement (1 to 10 score), and whether or not the signer identified the visualization that correctly matched the animation (1 = correct, 0 = incorrect). We will concisely refer to these categories as: "grammaticality," "understandability," "naturalness," and "match-success" (or "match percentage").

Average Results per Group

Figure 53 shows the average scores for grammaticality, understandability, and naturalness for each of the three groups: motion-capture animation, classifier predicate generator animations, and Signed English animations The classifier predicate generator's higher scores in all three categories is statistically significant ($\alpha = 0.05$).[59] As we expected, the classifier predicate generator outperformed the Signed English animations – which we had considered a lower-baseline for the system's performance.

What is surprising is that the CP animations also outperformed the motion-capture animations – which we had considered our upper baseline. It seems unlikely that our prototype classifier predicate generator is actually outperforming human ASL signers at

[59] Calculated using pairwise Mann-Whitney U tests with Bonferonni-corrected p-values.

producing grammatical, understandable, and natural ASL performances. More likely, the difficulties in recording the motion-capture performance from a human signer left too many errors and distortions in the data for us to successfully repair during the post-recording clean-up process. This reduced the quality of the motion-capture animations in such a way that our classifier predicate generator was able to beat it in the evaluation.

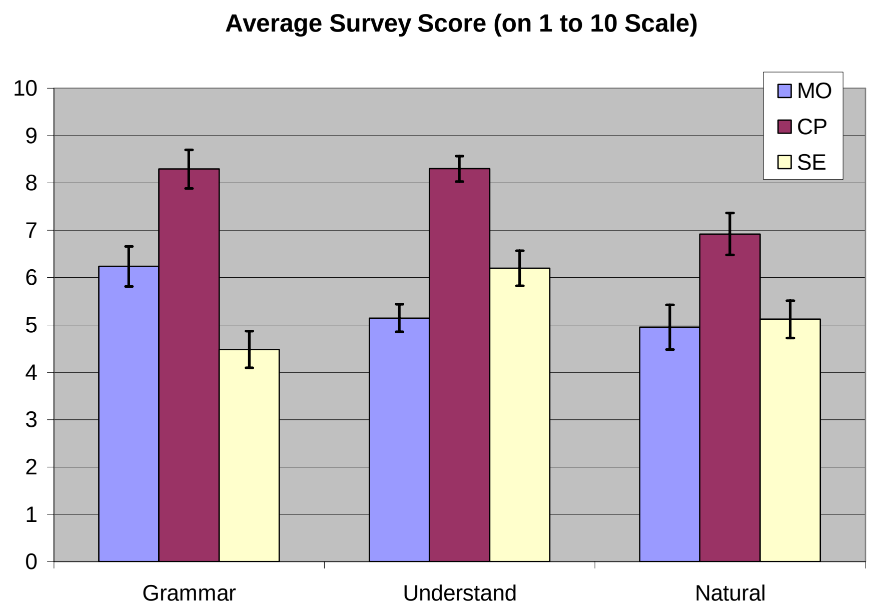


Figure 53: Average Grammaticality, Understandability, and Naturalness per Group

Although we are primarily interested in comparing the classifier predicate generator to the other two groups, we did notice some statistically significant differences between the motion-capture and Signed English animations in two categories. The motion-capture animations received significantly higher grammaticality scores than Signed English, yet Signed English received significantly higher understandability scores than motion-capture. Since the Signed English animations follow English grammar (and not ASL grammar), it is not surprising that the motion-capture animations received higher grammaticality scores. (The lower scores for Signed English provide some evidence that participants understood that they were asked to evaluate the *ASL* grammaticality of the animations – not their English grammaticality. However, it is somewhat surprising that the Signed English scores were as high as they were.) The fact that subjects rated the Signed English animations higher in "understandability" than the motion-capture animations may indicate that there were still errors and distortions in the motion-capture animations. It may also be due to some subjects' strong English skills – see Footnote 58.

Figure 54 shows the percentage of correct matches for each group. Once again, the scores for the classifier predicate generator animations are significantly higher than the motion-capture or Signed English animations. (The difference in match-success between the motion-capture and Signed English groups is not significant.) Just like the results for the grammaticality, understandability, and naturalness, our classifier predicate generator received higher scores than both its expected lower- and expected upper-baselines.

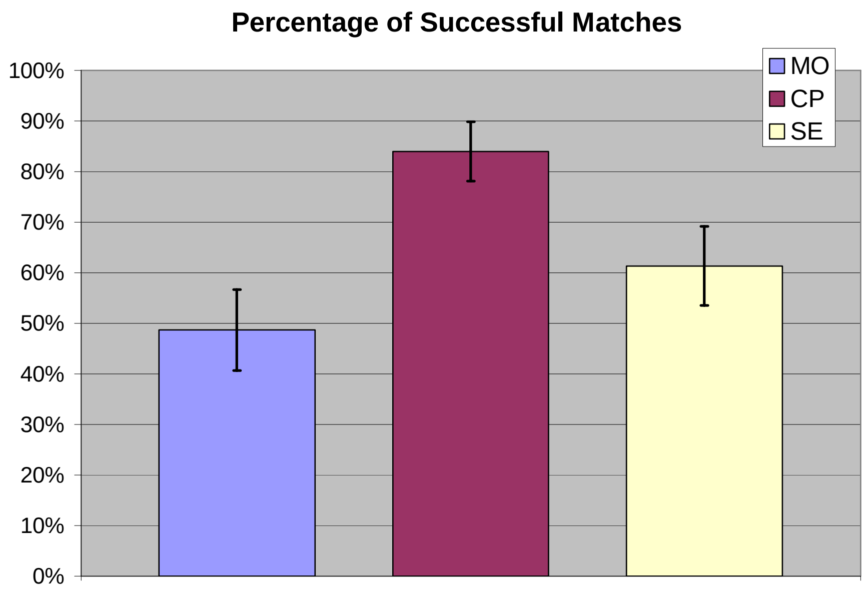


Figure 54: Match-Success Percentage per Group

It's clear that the motion-capture animations created for this study did not serve as an upper-baseline measurement. Without a set of motion-capture animation scores as our upper-baseline, we are forced to consider a perfect score for grammaticality, understandability, and naturalness, and match-success as our upper baseline. Even from this perspective, the performance of the classifier predicate generator in this study was quite good – especially for an initial prototype implementation of the system.

Table 9 lists the standard deviation in per-response scores for grammaticality, understandability, and naturalness. The first column contains the standard deviation of the per-response scores for all of the data, and the final three columns show the standard deviations for each group. The low standard deviations for grammaticality and understandability for the classifier predicate animations is of interest. This indicates that the CP generator's scores for these two categories were not only significantly better than the scores of the other two groups, but they were also composed of individual response scores that were more consistently closer to the mean.

Table 9: Standard Deviation of Grammaticality, Understandability, and Naturalness

	All Data	**MO only**	**CP only**	**SE only**
Grammaticality	2.95	2.63	1.79	2.95
Understandability	2.65	2.56	1.69	2.58
Naturalness	2.56	2.41	2.32	2.47

Average Results per Animation

This section will present the survey results on a per-animation basis – allowing us to explore the results for particular sentences that were included in the evaluation study.[60] Figure 55 shows average grammaticality scores for each animation. Across the ten sentences, the scores for the classifier predicate and Signed English animations remain fairly consistent, but the scores for motion-capture seems to vary more across sentences. For example, the grammaticality score for motion-capture sentence #10 was quite low.

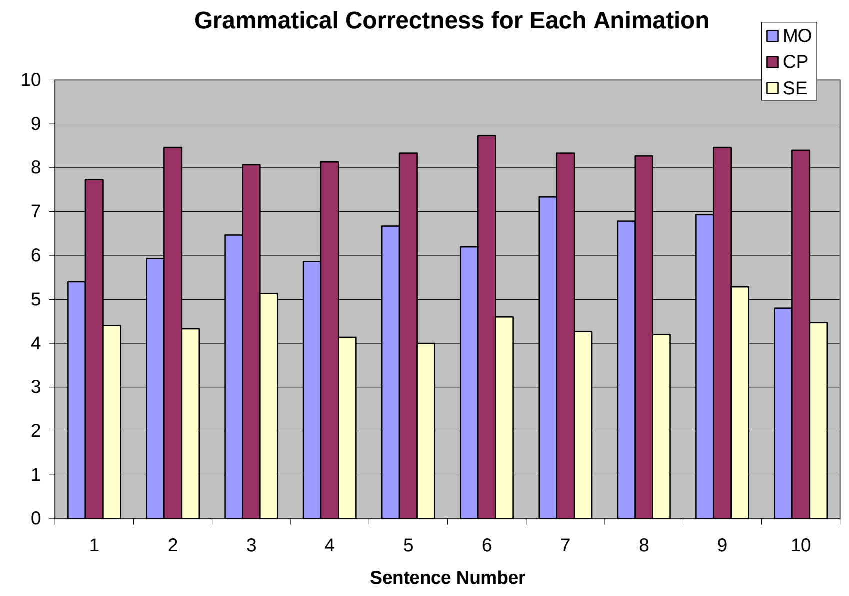


Figure 55: Average Grammaticality Scores for Each Animation

In the motion-capture animation for sentence #10 ("the woman is next to the table"), the human signer performed the following during the motion-capture recording process:

1. ASL sign "TABLE" is performed with both hands.

[60] There is too little data (n=15) to obtain statistical significance between the bars for each sentence in the graphs in Figure 55, Figure 56, Figure 59, and Figure 60. While we will consider the results on a per-animation basis in this section to explore some of the important factors in the evaluation, these graphs should not be taken to indicate statistically significant differences for any one particular animation.

2. Non-dominant hand in “Flat B” handshape shows location in space for the table.
3. ASL sign “WOMAN” is performed with dominant hand. (The non-dominant hand lowers to a resting position at this time.)
4. ASL sign “NEXT-TO” performed with both hands.

There are two reasons why this sentence may have received a low score. Some participants may have felt that there was an actual grammatical problem with the sentence as it was performed by the original human signer – in other words, even if we had shown them a video tape of the original signer, they may not have liked the performance. (After noticing this potential grammaticality problem with this animation, we realized that we would have liked to have been able to show a videotape of the original performance to our participants in order to confirm whether this was the case. Unfortunately, since we had not anticipated this need before the beginning of our study, videotaping our research subjects was not listed as part of our Institutional Review Board protocol.)

Aside from there potentially being a grammatical error in the ASL performance of the human wearing the motion-capture suit, another possible explanation for the low score for motion-capture sentence #10 is that the motion-capture process introduced errors in joint angles that reduced the quality of the animation in such a way that it became ungrammatical. For instance, the signer’s palm orientation in the animation is not quite horizontal during the performance of the “Flat B” classifier predicate showing the table’s location – the table is somewhat tilted forward. During the original performance of this ASL sentence, the human signer’s hand was actually horizontal.

Figure 56 shows the average understandability scores for each animation. Like the grammaticality scores, the understandability scores for the classifier predicate generator are fairly consistent across the ten sentences; however, the scores for the motion-capture and Signed English animations seem to vary more across sentences. To understand why some of the motion-capture animations received such low understandability scores (and why others did not), we shall examine one of the motion-capture animations in detail.

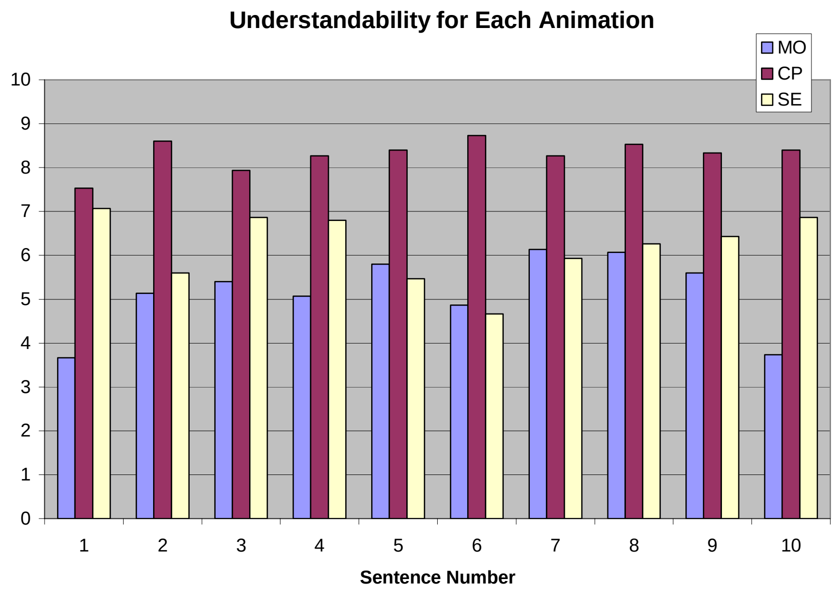


Figure 56: Average Understandability Scores for Each Animation

In sentence #1 ("the car parks between the house and the cat"), the human signer showed the cat's location, the car's initial location, and the house's location before showing the car's movement path. By showing the car location once and then showing its motion path at a later time in the performance, this animation was particularly susceptible to degradation in quality from the introduction of small errors in the joint angles from the motion-capture process. If the location that the arm initially indicated for the car does not match up properly with the second car classifier predicate showing the movement path, then it could be confusing as to how many vehicles are in the signing space. The two frames of the animation shown in Figure 57 show the location initially assigned to the car (with the signer's right hand) and the location the car started its motion path from later in the animation – these two locations don't quite align due to joint angle errors introduced during the motion-capture process.

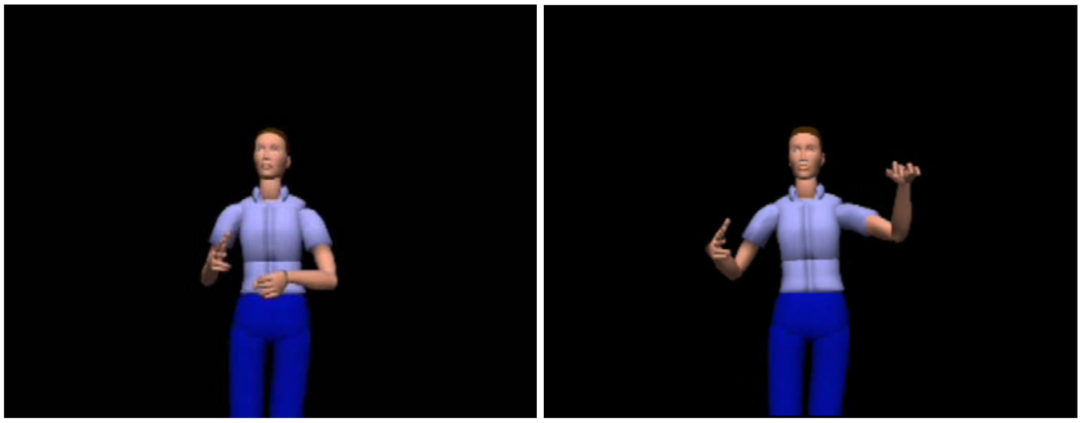


Figure 57: Two Frames from Motion-Capture Animation #1 – Car Locations

It is also somewhat difficult to see the ASL sign CAT in this animation. Signs in which the hands must make contact with part of the signer's body can be particularly difficult to record with the motion-capture system. If the joint angles are slightly off (or the body proportions of the virtual human character do not match the body proportions of the human signer that was recorded), then the hand might not correctly make contact with the part of the body that it should to correctly produce the ASL sign. As we can see in Figure 58, the locations of the hand during the sign CAT are too far to the signer's left. (The sign should be performed near the right cheek, not in front of the mouth.)

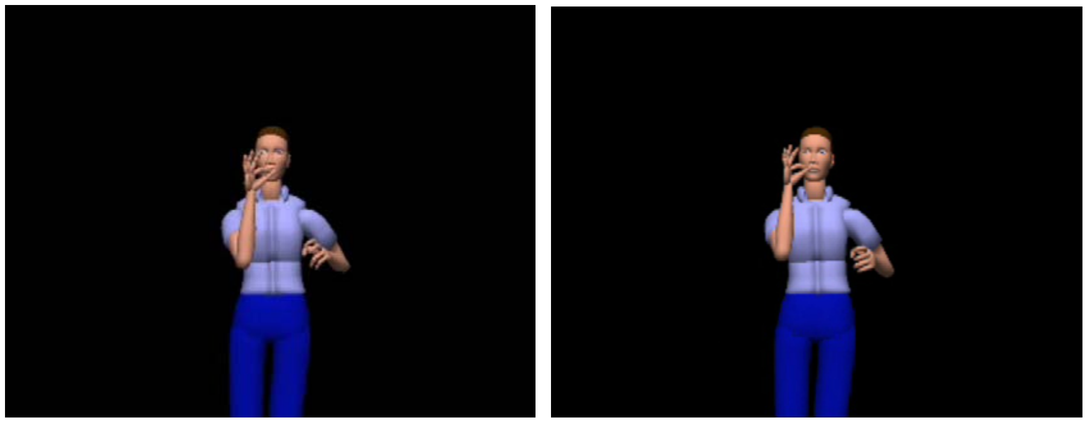


Figure 58: Two Frames from Motion-Capture Animation #1 – ASL Sign CAT

Thus, we can see how a minor error in the motion-capture recording of the signer's arm results in the hand reaching a slightly different part of the face, and this makes the ASL sign CAT somewhat difficult to understand in Motion-Capture Animation #1. The variety of ASL signs and constructions which occur in the recorded sentences makes some performances more sensitive to joint angle errors, and the variety of factors that affect motion-capture accuracy can make it hard to predict when errors will get introduced. While the motion-capture animations were edited after being recorded to apply transformations to the streams of joint angles (by adding/subtracting offsets and increasing/decreasing magnitudes of angles) to reduce the perceived distortion, it was sometimes difficult to find an appropriate correction. Since we wanted to base the animations on a real human performance and not on the talents of an animator, we did not want to over-edit the recordings to hand correct individual joint errors that appear during particular signs. The net effect of all of these factors is that some of the motion-capture animations used in this study are quite understandable, and others suffered errors that caused the animation to become difficult to understand. This helps to explain the variability in understandability scores for motion-capture animations in Figure 56.

Figure 59 shows the average naturalness scores for each animation. Similar to the graph for understandability scores, we can see that the classifier predicate animations have fairly consistent scores across sentences (generally staying around 6.5 or 7), but the motion-capture and Signed English animations show some greater variability across sentences. The occasional low scores for motion-capture sentences may be the result of "jittery" movements that sometimes occur in these animations.

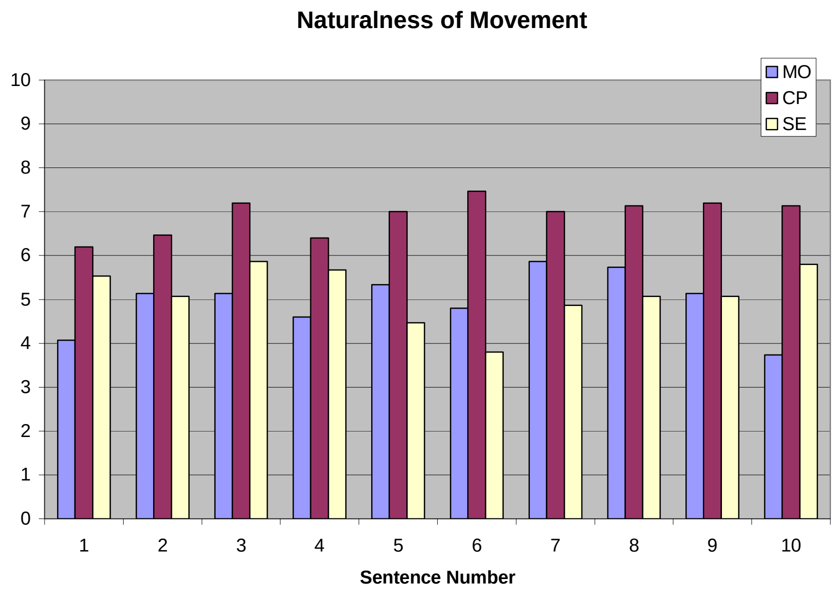


Figure 59: Average Naturalness of Movement Scores for Each Animation

Figure 60 shows the percentage of correct matches for each animation. This is percentage of times that a participant in the study selected the correct visualization for that animation. All of the groups show a greater variability in their values across sentences than we've seen in the previous graphs for grammaticality, understandability, and naturalness. This increased cross-sentence variability is partially due the introduction of a new factor: the set of three animated visualizations for each sentence. For this metric, the particulars of what each sentence is about (and how well each of the different animations successfully conveys the arrangement of objects in the visualization) becomes important to the result. Therefore, it's understandable that the results across sentences will show more variability.

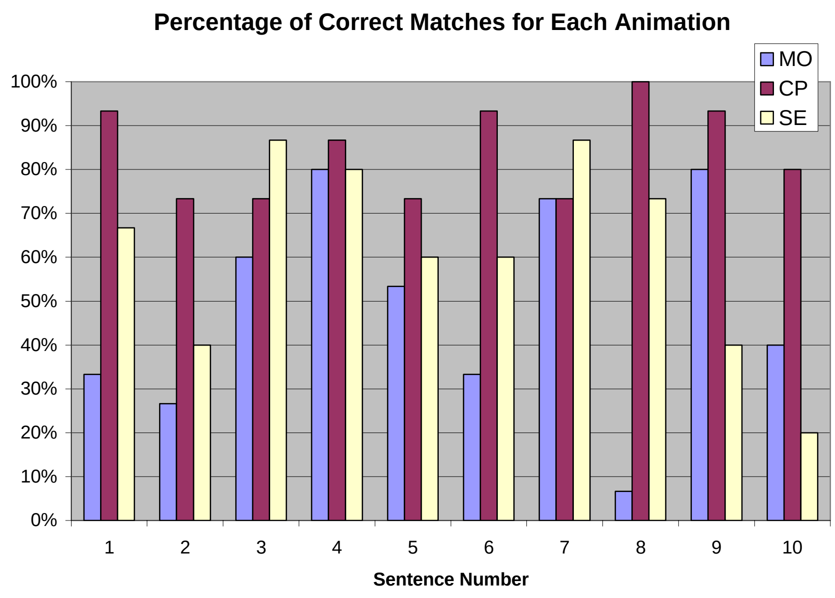


Figure 60: Percentage of Correct Matches for Each Animation

An interesting feature of this graph is that for sentences #3 and #7, the Signed English animations received the highest percentage of correct matches.[61] For sentence #3 ("the car turns left"), one of the confusables was a visualization in which the car drives forward and turns to the left as it skids to a stop. In the correct visualization, the car continues to drive for a short distance after it turns left. Some respondents felt that the

[61] While this graph summarizes the data recorded in the study, the differences in scores for individual sentences are not statistically significant.

motion-capture and classifier predicate animations were describing the first confusable. (This correct/confusable difference was one of the more subtle ones in the study.)

In sentence #7, there was another pair of similar visualizations. The correct visualization showed a man walking away from a woman along a diagonal path toward the left foreground of the animation. One of the confusable animations showed man walking past a woman – his motion path began somewhat behind the woman, and he also moved to the left foreground of the animation. The first animation frame for each of these visualizations is shown in Figure 61 with a white arrow drawn on each image to indicate the man's motion path.

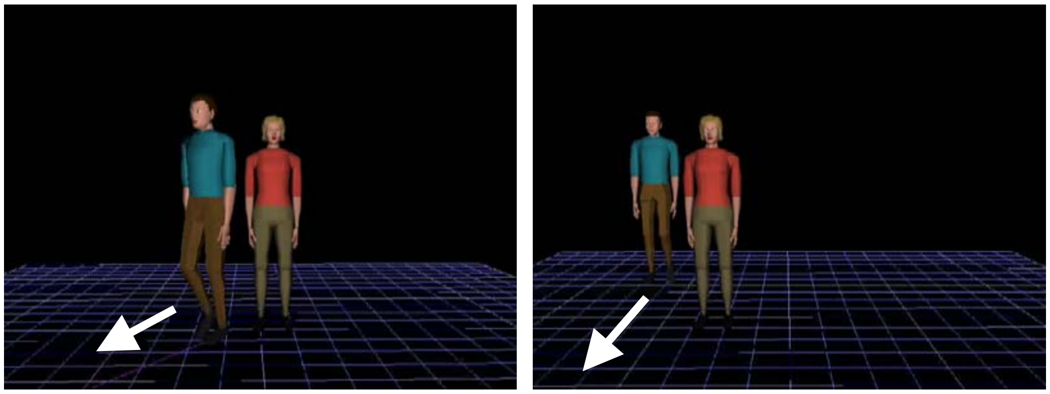


Figure 61: Correct Visualization and One of the Confusables for Sentence #7

The essential difference between these two visualizations is the "depth" in the 3D animation at which the man begins his motion path. After watching a classifier predicate or motion-capture animation, some respondents may have selected the incorrect visualization because they could not determine the "depth" at which the signer's hand began its motion path when performing the classifier predicate describing the man's motion. Small differences in the depth of the signer's hand in the signing space are sometimes difficult to see (since the animation is displayed on a two-dimensional screen).

Another explanation for the Signed English animation's higher score could be the presence of the sign "FROM" in the Signed English sentence. This lexical item may have emphasized the importance of the *beginning* of the man's motion path and helped the respondent select the correct visualization. Of the three options for sentence #7, it was only in the correct visualization that the man's motion path began near the woman.

Perceived vs. Actual Understanding

Across the 450 responses in this data set, there are significant (α = 0.05) pairwise correlations between grammaticality, understandability, naturalness, and match-success. The correlation coefficients (Pearson's R-values) are given in Table 10; we can see that there are moderate correlations between the grammaticality, understandability, and naturalness scores. This was not surprising since the grammaticality and naturalness-of-movement of an animation could certainly affect how understandable it is. A more surprising result was how weakly correlated the match-success is to the other scores.

Table 10: Pairwise Correlation Coefficients for the Four Categories (All Data)

	Grammaticality	**Understandability**	**Naturalness**
Understandability	0.63		
Naturalness	0.62	0.70	
Match-Success	0.09	0.21	0.12

While match-success is most closely correlated with understandability, we would have expected that the respondent's perception of the understandability of an animation to be more closely correlated with her actual success at selecting the proper visualization. If we partition the 450 survey responses into groups based on the understandability score of that response and we calculate the match-success percentage of each group, then we obtain the graph in Figure 62. While there is a relationship, the understandability score that a respondent selects is not a particularly strong indicator of whether or not she will select the proper visualization (i.e. whether or not she understood the spatial information conveyed by the sentence). This suggests a difference between a respondent's "perceived understanding" and her "actual understanding" of an animation. This analysis suggests that a respondent's reported scores for understandability are no substitute for the actual visualization matching data. Without the match-success values collected in this evaluation study, we would not have been able to determine whether respondents actually understood each animation.

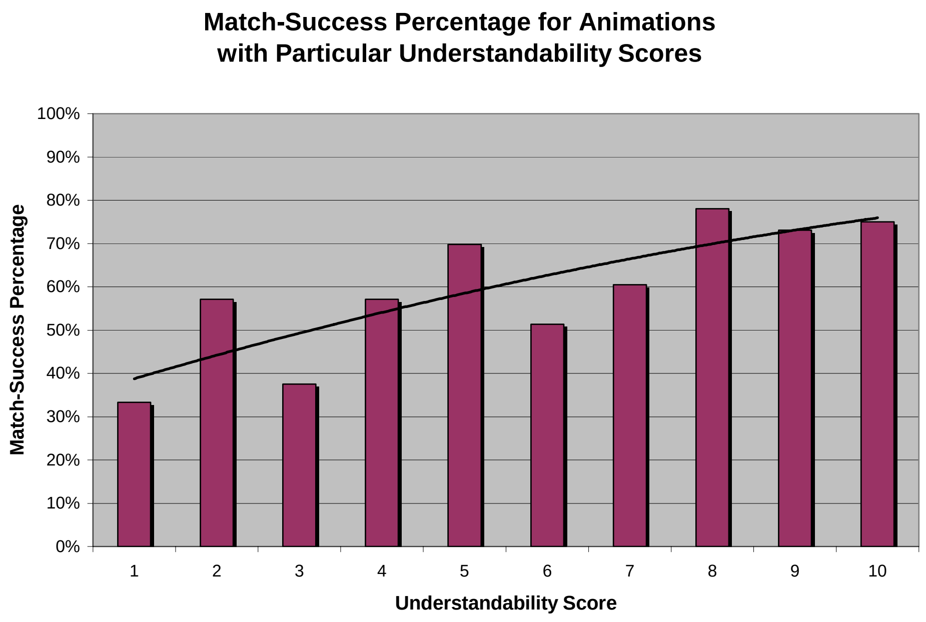


Figure 62: Match-Success Percentage for Responses with Particular Understandability Scores

The graph in Figure 62 shows data from all three groups: motion-capture, classifier predicate generator, and Signed English. Since the Signed English animations are different in structure than the classifier predicates generated by the system or collected through motion-capture, then it may not be appropriate to combine all three groups of data when considering this correlation. Table 11 and Figure 63 are based on data from the motion-capture and classifier predicate generator animations only – showing a similar pattern of results to those in Table 10 and Figure 62.

Table 11: Pairwise Correlation Coefficients for the Four Categories (SE Data Excluded)

	Grammaticality	**Understandability**	**Naturalness**
Understandability	0.81		
Naturalness	0.68	0.70	
Match-Success	0.19	0.30	0.19

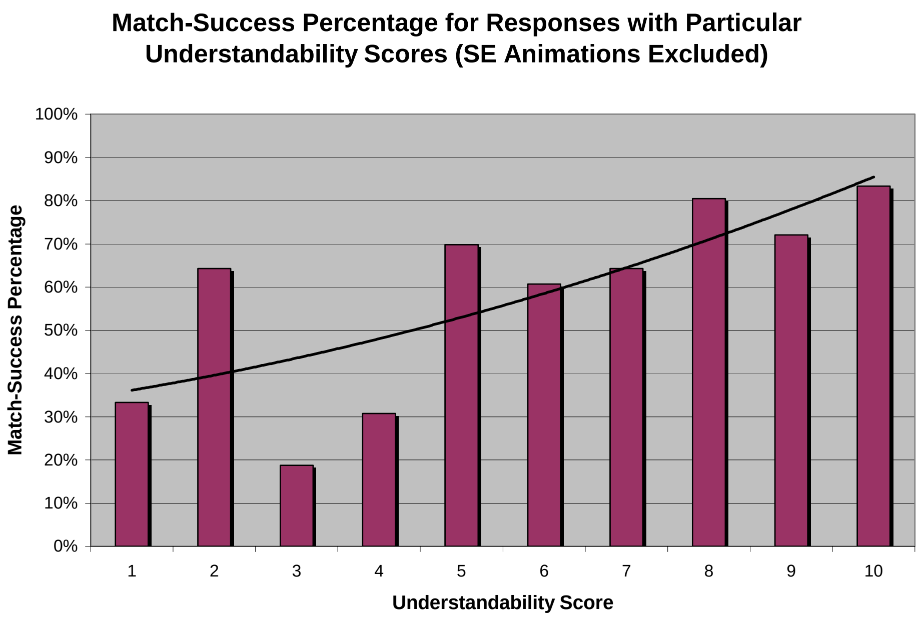


Figure 63: Match-Success Percentage for Responses with Particular Understandability Scores (SE Animation Data Excluded)

Omitting the Lowest-Quality Motion-Capture Animations

Throughout this chapter, we have discussed how errors were introduced to the joint angle data during the motion-capture process and how the resulting distortions led to some motion-capture animations receiving lower evaluation scores. We've also discussed how the particulars of the ASL sentence being performed can make the scores of some of motion-capture animations more sensitive to these joint angle errors. We had originally intended the motion-capture animations to serve as an upper baseline in this study, and so it would be interesting to see how much the evaluation scores would rise if we omit some of lower-quality motion-capture animations from the study. We've discussed problems with motion-capture animations #1, #4, and #10 earlier in this chapter, and there is one more animation that has several noticeable errors from the motion-capture process: animation #6 ("the man walked between the tent and the frog"). This animation suffers from some of the same hand location inconsistency and sign-production error problems as we found in animation #1 (see page 245). These four animations received the lowest understandability scores in the evaluation study.

When we omit the evaluation-survey data from motion-capture animations #1, #4, #6, and #10, then we produce the graphs in Figure 64 and Figure 65. Despite the higher motion-capture scores (and smaller sample size), the CP animations continue to score significantly ($\alpha = 0.05$) higher than motion-capture and Signed English in all categories.

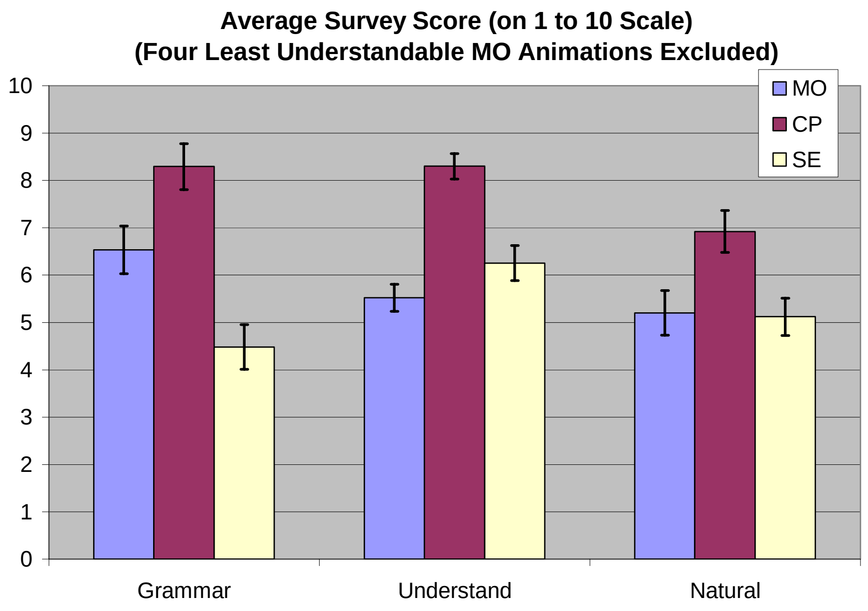


Figure 64: Survey Score Data After Removing 4 Bad Motion-Capture Animations

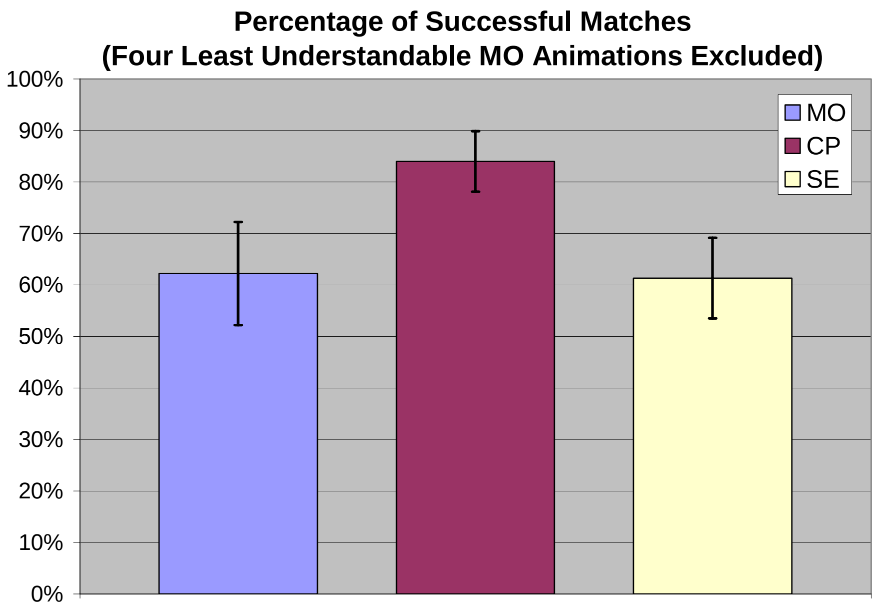


Figure 65: Correct Matches After Removing 4 Bad Motion-Capture Animations

After omitting these four motion-capture animations from the study, the only change with regard to statistical significance is that the Signed English animations are no longer significantly better than the motion-capture animations in terms of understandability.

Should We Penalize Signed English Animations for Ambiguity?

Given the information contained in a Signed English animation, it is sometimes ambiguous which of the three visualization options should be the correct match for a sentence. Unlike the 3D information encoded in a classifier predicate, the information in Signed English sentences is sometimes not enough to disambiguate between the visualizations. For example, given the sentence "THE WOMAN STANDS NEXT-TO THE TABLE," it would not be possible to determine which of the three visualizations listed in Table 6 (page 225) is being described. If a respondent in the evaluation study did not select the correct visualization for this sentence, the response will be recorded in the data set as having the value "false" for its "match-success" category. It may seem strange to penalize the Signed English animation for this – after all, any of the three visualizations for this sentence is consistent with the Signed English sentence "THE WOMAN STAND NEXT-TO THE TABLE." Even if the respondent saw a human signer performing this Signed English sentence, he would not be able to determine which of the visualizations is correct.

Under a more "lenient" scoring guideline, we could consider any of the three visualizations a correct choice for the Signed English animation. Specifically, for each sentence, we could decide how many of the visualization options are consistent with the information contained inside the Signed English animation, and we could record any of these responses as a correct match. The match-success graph that would result is shown in Figure 66. (Any of the three visualizations for sentence #10 is accepted as correct, and two visualizations for sentences #2, #6, #8, and #9 are considered correct.)

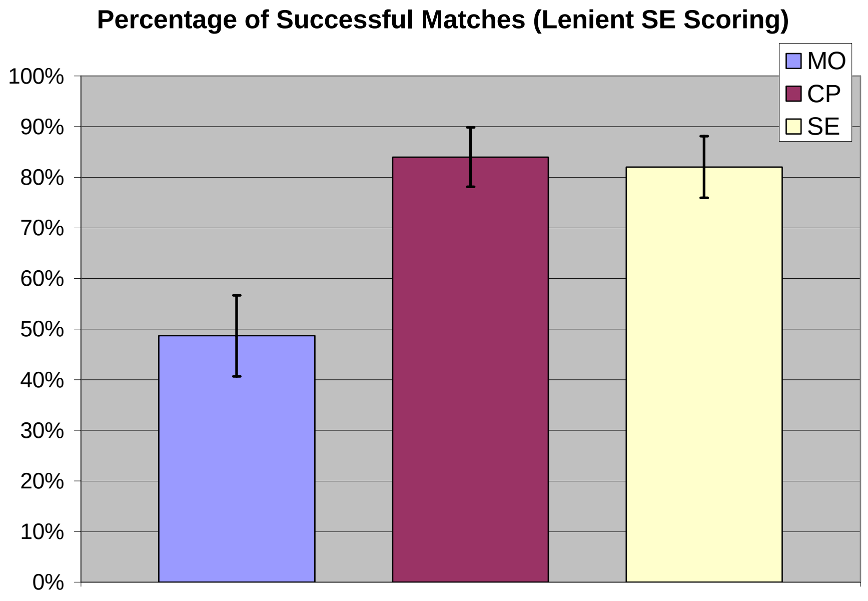


Figure 66: Match-Success for Each Group with "Lenient" Signed English Scoring

Under this “lenient” scoring guideline, there would no longer be a significant difference between the match-success scores for classifier predicate and Signed English animations. While it is interesting to consider the effect of this scoring system, there is a problem with this approach. By holding the Signed English animations to this lesser standard, we are artificially boosting its match-success score, and we have bypassed a limitation of Signed English sentences *that should be included in the evaluation*.

It is an essential characteristic of Signed English sentences (with their use of spatial prepositions – like English sentences) that spatial relationships are sometimes left ambiguous in a way that they are not by ASL classifier predicates. Signed English sentences do not include the specific 3D information conveyed by classifier predicates, and the English proficiency of the person viewing the animation becomes an important factor in his or her ability to successfully understand the message being conveyed. (Specifically, the person must understand the semantics of the spatial prepositions, and he or she may need to resolve ambiguities in their spatial meaning using contextual information or world-knowledge.) Since we are conducting this study to evaluate a classifier predicate generator designed for use in an English-to-ASL MT system (for users with limited English literacy), we don't want to leave this kind of processing burden on the person viewing the animation output.

While it might seem unfair to penalize a respondent in our study for selecting an "incorrect" visualization even when there was not enough information in the Signed English animation to make the decision, this is the way in which we penalize Signed English animations as a group for leaving this spatial information ambiguous. It is not the individual respondent's skill nor the particular Signed English animation that were are interested in measuring with the match-success metric. We would like the scores we obtain for match-success to represent how successful a whole group of animations would be at conveying information as the output of an English-to-ASL MT system. This is why the "strict" scoring guideline was used in this evaluation study.

Qualitative Feedback from Participants

At the end of the evaluation study, participants were shown three classifier predicate animations: #1 ("the car parks between the house and the cat"), #10 ("the woman stands next to the table"), and #6 ("the man walks between the tent and the frog"). They were asked to comment on several aspects of each animation: the speed of the animation, its color and lighting, the visibility of the signer's hands, the correctness of the hands' movements, the correctness of the signer's facial expressions, and the correctness of the signer's eye-gaze. Each participant was also invited to give specific recommendations on ways to improve the animations.

Of the fifteen participants, eight mentioned that some of the animations were a little too slow, and one participant felt that they were very slow. Eight participants also mentioned that the animations should have more facial expressions, and four of these specifically mentioned that the animations were missing nose and mouth movements. Four participants mentioned that the signer's body should appear more loose/relaxed and that it should move more during the performance. Two participants wanted the signer to convey more emotion during the performance. Two participants felt that the eye-brows should go higher when they are raised, and three participants felt that there should be more eye-gaze movements during the animation. Two participants felt that the blue color of the signer's clothing was a little too bright, and one disliked the black background.

Some participants commented on the performance of particular ASL lexical signs that they thought were performed incorrectly. Three participants discussed the ASL sign "FROG": one felt it should be performed a little more to the right of its current location, and another felt that the hand should be oriented with the fingers aimed more to the front. One signer felt that the way in which the thumb and forefinger touched as part of the handshape of the ASL sign "CAT" should be slightly different. Another signer felt that the ASL sign "TABLE" was performed a little too high in the signing space.

Some participants commented on the classifier predicate portions of the performance. For sentence #1, one signer felt that it would be better to use the non-dominant hand to hold the location of the house during the car's movement (instead of using the non-dominant hand to create a platform for the car to park upon). However, another signer commented that it was important to make sure that the dominant hand actually makes contact with the non-dominant "platform" hand at the end of the car's movement path. For sentence #6, two signers suggested that instead of using a classifier predicate with a "Spread C" handshape to show the location of the tent, the signer could simply perform the ASL sign "TENT" at the location in space where it is located. Some ASL noun signs can be performed at locations in the signing space in order to set up an object at that location (MacLaughlin, 1997; Liddel, 2003a). This capability is currently beyond the scope of our classifier predicate generator.

Demonstrating Key Properties of the System

Unlike an evaluation of a broad-coverage NLP system, during which we obtain performance statistics for the system as it carries out a linguistic task on a large corpus or "test set," this chapter has described an evaluation of a prototype NLP system. A key difference is that this study was not designed to test the linguistic coverage of the system – our prototype system generates ASL animations from a set of sample inputs loaded into a lookup table in the system's memory. So, what exactly has this study tested?

After implementing the prototype system and successfully processing a few sample inputs stored in the lookup table, we already knew that we could produce a Partition/Constitute tree that represented an animation performance. We also knew that we could process this P/C tree to create an animation of the movements of a virtual human character. What we did not know was whether the system was actually able to produce animations that were grammatical ASL classifier predicates that conveyed the information in the input sentence.

Any one of these design features below could have been a limiting factor that would have restricted the space of possible animations the system can produce from including grammatical ASL classifier predicates (any of these could have been the weak link in the chain of processing steps and representations on our way to producing an ASL output):

- the way we modeled 3D scene visualization data,
- the way we encoded English Predicate Argument Structure,
- the information stored inside the Discourse Model,
- the use of hierarchical planning operators to encode inter- and intra-classifier-predicate structure,
- the use of P/C trees to represent the temporal structure of an ASL performance,
- the representation of the body movement as a set of articulators that contain dynamic locations, orientations, handshape, and Boolean values,
- the use of spatial transformation functions to specify these values,
- the inverse kinematics arm/hand motion-planning algorithm,
- or the representation of face and eyes using the Greta facial model.

The high scores for grammaticality and match success that native ASL signers gave to the animations produced by the classifier predicate generator suggest that these animations were grammatical ASL classifier predicates and that they conveyed the proper 3D spatial information. Thus, in terms of the list of “key properties” introduced in Chapter 1, this study has provided evidence that the classifier predicate generator is ***functional.***

What We Learned about the Design of ASL Evaluation Studies

From the comments that subjects gave in the qualitative portion of the evaluation study, we could tell that they were comfortable critiquing the animations, and most of them were willing to suggest specific elements of the performance that should be changed in order to improve the animation. In fact, some of the recommendations were quite detailed – some participants commented on slight differences in the location or orientation of the hand or the degree to which the eyebrows of the signer were raised. Their comments suggested modifications that we can make to the system – and we can test these modifications in future evaluation studies (see Chapter 10). The quality of their feedback suggests that any future evaluation study also include a qualitative portion.

The low correlation we observed between the understandability score that a participant gave to an animation and his or her actual success at selecting the proper visualization for that animation calls into question whether “perceived understandability” scores are actually indicating how much “actual understanding” the participant had. This suggests that we should include some form of comprehension task (whether it be a matching task or some other form) in any future evaluation study of ASL animations. The use of understandability scores is no substitute for this kind of data.

We also saw that the introduction of errors in the human joint angle data collected during motion-capture can reduce the quality of these animations such that they do not serve as an upper-baseline in the study. The development of a better upper-baseline (either one that is produced in a different way or one that is produced through this motion-capture approach but with fewer errors) will be explored in future work. Chapter 10 will discuss alternate ways of developing an upper-baseline.

Although the set of motion-capture animations used in this study did not serve as an upper-baseline, our experiences with them have shed light on the difficulty of the ASL generation task. Specifically, the low scores that participants gave to these animations illustrate how accurate we must be when controlling a virtual human character to perform ASL animations. If we assume that the human ASL signer (who wore the motion-capture suit) performed grammatical, understandable, and natural-moving ASL sentences, then the poor evaluation scores for the motion-capture animations illustrates the impact of errors introduced during the motion-capture process – specifically, small errors in the joint angles of the animated character's body. This indicates that for some ASL sentences, having joint angles that are *approximately* the right value is not sufficient – especially since a small error in each joint of the clavicle/shoulder/elbow/wrist can have a cumulative effect of positioning the hand in the wrong location or orientation.

Our classifier predicate generator plans the ASL animation by specifying a location and orientation for the hand, and then it uses inverse kinematics to decide the angles for the shoulder, elbow, and wrist that will get the hand into this position. The evaluation results for the motion-capture animations helps to illustrate why this was a good design choice (and was a better approach than attempting to directly specify all of the joint angles of the arm during the classifier predicate planning process). By designing the generator to start with the hand location/orientation and "work backwards" to determine the joints for the rest of the arm, we have avoided the effect we noticed in the motion-capture animations: the accumulation of small joint errors to produce a larger error in the hand location/orientation.

Chapter 10:
Future Work

This chapter will suggest future modifications to the classifier predicate generator, and it will outline the further development of the English-to-ASL MT system. This chapter will also describe how future user-based evaluation studies can be designed to evaluate and direct the course of development.

Development of an English-to-ASL MT System

This dissertation has included a design for the overall architecture of an English-to-ASL MT system, and a prototype of the classifier predicate generator (the most novel component of this MT design) has already been implemented. This prototype includes the entire processing architecture of the generator, but it only contains an initial set of classifier predicate templates. As discussed in the previous two chapters, the implementation and evaluation of this prototype demonstrated that the generator can be implemented in a practical, scalable manner and that native ASL signers find the output grammatical and understandable.

The next stage of development for this system is to implement (and evaluate) more of the English-to-ASL architecture so that the system has broader linguistic coverage. This work consists of two parts:

- Implementing generation components that produce animations of ASL sentences that do not contain classifier predicates.
- Extending the coverage of the classifier predicate generation component and making modifications to its design.

Future Development of the Non-CP Components

Chapter 5 of this dissertation discussed the multi-path English-to-ASL MT architecture, and it also included a discussion of the Lexical Signing and Signed English pathways of this design. This chapter included two sections that highlighted some issues that are important for the development of these non-classifier-predicate parts of the MT system: “How We Could Implement the SE Pathway” on page 104 and “How We Could Implement the LS Pathway” on page 105. The discussion in these sections gives some additional detail about the future implementation of these parts of the system.

Much like the hand-built Signed English animations produced for the evaluation study, it is anticipated that a Signed English component in the MT system would assemble a set of signs together (from a library of animation files) into a sequence that is performed by the virtual human character. The system would use a simple dictionary-lookup process to select the appropriate sign, and it may need to incorporate word sense disambiguation technology in the English analysis component to select the proper sign.

The selection of a specific translation and generation approach for the Lexical Signing pathway is left to future work, but it is anticipated that some of the technology from previous English-to-ASL MT systems (that have focused exclusively on this type of ASL signing) could serve as a starting point for this design. While many design choices will need to be made for this pathway, there are some interesting issues surrounding the representation of the signing space and the implementation of certain verbs that we will consider below.

Chapter 5 discussed some issues surrounding the use of space during Lexical Signing animation for pronoun reference and verb agreement. Similar to the "Ghosts" in the "Depicting Canvas" used for classifier predicates, the Lexical Signing pathway of the system will use a model called a "Token Canvas" containing "Token" objects that represent locations in space associated with entities under discussion. Unlike Ghosts, these Tokens would not include 3D orientation information, and their location in space would not be used to represent the topological layout of a 3D scene. Chapter 5 outlined some factors that would need to be considered when determining how to arrange these Token objects in the signing space.

The way in which the classifier predicate portion of the system used spatially parameterized templates that specify articulator values using spatial transformation functions may inspire the design of ASL lexicon entries for non-classifier-predicate signs that also undergo movement changes. For instance, ASL "agreement" verbs (whose motion path or orientation can be modified to indicate locations in the signing space associated with the subject and/or object) could be implemented as templates that use a spatial transformation function (for that particular verb's motion path) that accepts two parameters: the location of the Token for the verb's subject and the location of the Token for the verb's object. The function would calculate a direction/distortion/stretching of the movement of the verb to indicate the subject and object locations.

Chapter 2 also discussed a class of ASL verbs called "spatial" or "depicting" verbs which can indicate topologically meaningful motion paths as part of their performance. To implement these verbs in the Lexical Signing pathway, the system may need to allow this pathway to also access a Depicting Canvas representing the layout Ghosts for a 3D scene. A template specified for this type of verb could use Ghost locations/orientations as the parameters to a spatial transformation function that calculates the verbs motion path or orientation during the performance.

As part of the implementation of the Lexical Signing pathway, we will explore how to best represent the signer's head-tilt, shoulder-tilt (for use in ASL "role-shifting constructions"), and other non-manual aspects of the performance as a set of articulators with values similar to those we have used for the hands, brow-raise, and eye-gaze. We expect that ASL linguistic research on the phonological representation of this type of information will prove useful when designing this portion of the system.

With the development of these additional pathways of the system, we can begin to explore whether the "fall-back" approach (introduced in Chapter 5) is sufficient for selecting the pathway of the English-to-ASL MT system. If not, then we will explore alternate methods for selecting the proper form of signing output during English-to-ASL MT as part of this future work.

Chapter 8 discussed how the prototype classifier predicate generator used a lookup-table that included an English Predicate Argument Structure and Discourse Model entries for each sentence. As part of the development of a complete English-to-ASL MT system, English NLP technologies will need to be selected and customized to produce this type of information from an input English text string.

Future Development of the Classifier Predicate Generator

While much of the actual design of the Lexical Signing and Signed English pathways has been left to future work, we have a much more specific vision for how to continue the development of the classifier predicate portion of the system. The prototype classifier predicate generator built for the dissertation includes the processing engine and a small initial set of planning-operator "templates," each of which specifies one type of classifier predicate in the system's repertoire. To extend the linguistic coverage of this component, this set of templates will be expanded – a process which is scalable since the templates are built from a set of reusable animation subcomponents.

A broad-coverage implementation of the classifier predicate generator is a process that will require ASL-knowledgeable linguistic developers to create new classifier predicate templates. The use of spatial transformation functions in the design of these templates will isolate these developers from the specific 3D animation calculations that will be performed to create an animation. Instead, these developers can design the classifier predicate templates symbolically by linking the values of the articulators to values of objects in the linguistic models of the system.

Since this development process involves the creation of a large linguistic resource – the library of classifier predicate templates – we expect the increase in coverage of the classifier predicate generator to be an incremental process. As discussed in the

"Interleaving Core and Application Development" section later in this chapter, we anticipate creating assistive technology applications that make use of the ASL generation technology throughout the course of its development. We will make use of the incrementally increasing linguistic coverage of the system for each application. The applications that we select will help to determine which domain of classifier predicate templates should be the next focus of development along the way.

In addition to increasing the generator's linguistic coverage, we may also make further modification to the design of the generator that would affect any animation that it produces (even those we have already produced for the prototype version of the system). Based on the comments from participants in the evaluation study, we may explore increasing the speed of the classifier predicate animations, changing the height of the eye-brow raise of the signer, and adjusting the color of the signer's clothing. We will also explore ways to represent additional non-manual elements of classifier predicate performance as other articulators in the system. For instance, a classifier predicate showing the location of objects at different distances may use the signer's mouth to indicate close-distance, middle-distance, or far-distance with three different mouth shapes. These alternatives would need to be represented in the system as an articulator with a value that could be specified during the classifier predicate planning process.

As discussed in Chapter 8, the classifier predicate generator currently lacks a linguistically sophisticated mechanism for determining when to use the non-dominant hand to indicate the location of a surrounding object in the signing space and when to hold the location of this sign during the performance of a classifier predicate with the dominant hand. The production of these "secondary object" classifier predicate constructions will be another area of future work. The current implementation of the system may sometimes produce ASL classifier predicate constructions that are ungrammatical (or at least less fluent than they could be) since it does not generate these two-handed constructions.

Chapter 8 also discussed how – for the purposes of the prototype system – we stored a Transformation Matrix in the lookup table for each English sentence. This object specifies how to rotate, rescale, and recenter the Visualization Scene so that the objects it contains can be mapped onto the space in front of the signer's body. As part of future work, we will consider ways of selecting how to map the scene onto the signing space. Possible factors to consider may be the comfort of the signer in reaching the locations of the objects or in performing classifier predicates that indicate their motion paths. Various ASL linguistic factors governing the use of the depicting space around the signer (Liddell, 2003b) will also be considered.

For instance, a 3D scene could be mapped onto the signing space using several different scales and perspectives based on what information the signer wants to emphasize. To convey the sentence "the car drove up to the house," the signer could show a car and a house from a "zoomed-out" perspective by placing them both on an imaginary plane in front of their body using CPMLs with "Number 3" and "Spread C" handshapes. Alternatively, the signer could use one hand in a "Flat B" handshape to show a wall of the house and then use their other hand in a "Number 3" handshape to show how the car pulls up close to the wall. In this case, the signer may wish to emphasize how close the car came to the wall of the house. The selection of scale and perspective to convey these subtleties of meaning are left for future work.

Conducting Future Evaluation Studies

During the development of the classifier predicate generator (and of other portions of the system), it will likely be useful to conduct additional evaluation studies that focus on aspects of the design we are developing, new sets of classifier predicate templates, or new ASL phenomena produced by the Lexical Signing pathway. These studies would make use of animation baselines like those used in the evaluation of the prototype classifier predicate generator; however, Chapter 9 discussed several difficulties with the motion-capture animations that had been intended to serve as an upper-baseline. An important task for future work will be the creation of a better upper baseline for comparison.

The first option would be to refine the current motion-capture process using the ReActor system and CyberGloves. It is possible that further experience recording ASL signers using the system will allow us to improve the accuracy of the motion-capture by more-precisely placing the sensors on the signer's body, becoming more efficient at calibrating the CyberGloves, and spending time to more-accurately simulate the body proportions of the human signer with the virtual human character. We may also consider whether other forms of motion-capture technology (systems based on gyroscopic sensors, mechanical systems with exoskeletons measuring joint angles, etc.) could produce motion-capture data that is more accurate with less calibration time and post-processing.

Aside from motion-capture technology, there are other ways in which the performance of a human signer could be displayed as an option during the evaluation study. For example, we could show an actual videotape of the performance; however, we would prefer a baseline that allows us to compare the movement of a human performer but would still visibly resemble the virtual human character. This way we would know that the difference between the baseline and the system was due to differences in movement between the two – and not just the visual appearance. Another disadvantage of video is that it would be clear to the evaluation study participants when they were viewing a computer animation and when they were viewing a person's movements.

Another option would be to begin with motion-capture data and then perform more drastic post-processing modification to the data than was done for the animations in Chapter 9. We applied some transformations to the joint angles of those animations to add/subtract offsets or increase/decrease the magnitude of variation of particular joints; however, we tried to avoid directly scripting the movements of the character. We wanted to preserve the performance data as it was recorded from the human signer as much as possible so that the baseline remained "real" human movements. We could abandon this and instead adjust various portions of the animations in an ad-hoc manner to re-create the performance – we could videotape the signer during the motion capture session to help with this process. To confirm that the animation preserves the performance, we could also ask a native ASL signer (perhaps the one who performed the sentence in the videotape) to be present while the animation was adjusted to verify its accuracy.

There are also various evaluation studies that could be performed without the need for a baseline animation. For instance, if we are trying to select between several different options for modifying the system, we could run an evaluation study in which the different alternatives are compared against each other. This could be useful when modifying the classifier predicate generator to address some of the comments of participants in the prototype system's evaluation study. We could produce classifier predicate animations with different signing speeds, different degrees of eye-brow raising, different clothing colors, etc. Then we could use an evaluation study (with survey questions and the visualization-matching task) to determine which alternative is preferred.

We could also run an evaluation study before some of these alternatives are actually implemented in the system. We could produce an animation of how the system currently works, and then we could modify this original animation by hand to simulate the effect of the change we are considering making to the system. We could determine if there is a difference in performance, and we could use the results to decide if the change we are considering produces a benefit and whether the amount of benefit justifies the implementation effort.

Interleaving Core and Application Development

One of the initial motivations for the development of ASL generation and machine translation technology was to develop assistive technology applications for the deaf. While a broad-coverage English-to-ASL MT system would clearly be a useful tool for deaf users with low literacy skills, there are also many applications for ASL generation and MT technology prior to reaching this level of development. Even while ASL software is still limited to particular domains of text or is still limited in the types of ASL linguistic phenomena it can produce, this software will be useful for certain assistive technology applications.[62]

[62] Even subsets of the components of an ASL generation system actually have stand-alone applications that are useful from a deaf-accessibility perspective. For example, the portions of the English-to-ASL system that represent the use of space around the signer, represent the movements of the signer's body, and produce an animation output from these representations could be combined to create ASL scripting/generation software. Instead of translating an English source text into an ASL animation output, this software could be used by an ASL signer to "word process" an ASL animation performance that could be saved, edited, and played back as an animation at a later time. ASL lacks a standard writing system, and

Some educational applications for the deaf may only need to generate ASL signing to convey informational content within a limited domain. For instance, a tutoring program could include ASL animations that give feedback to the student; the system may need to produce combinatorially too many unique responses to videotape, but a simple generation system could be used to construct animations. As another example, we could create a tool for students learning ASL in which the system demonstrates the generation of classifier predicates from an arrangement of objects specified by the user.

We can study how to best embed the ASL technology into these applications and what the specific linguistic requirements on the software are for each application. As the ASL software is able to translate a wider variety of input sentences and/or generate a wider variety of ASL linguistic phenomena, then the set of applications in which it can be used will continue to grow. Along the way, new assistive technology applications will serve as research vehicles for improvements in MT software, and the particular requirements of each application will drive and prioritize what linguistic components of the ASL software should be the focus of development. Thus, we can interleave the development of the core ASL generation/translation technology with the creation of useful assistive technology applications.

---

so signers often record ASL performance using video. This software could give signers another option for creating presentations, literature, and messages in ASL, and it could have a substantial impact on the way in which students learning ASL practice and develop their skills.

Chapter 11: Discussion

This chapter will summarize the thesis of this dissertation, and it will highlight the major contributions of this work. This chapter will also trace how the special requirements of ASL generation have motivated the novel technologies in the classifier predicate generator and the English-to-ASL MT design, and it will describe how several of these techniques have applications beyond the generation of sign language.

Thesis of this Dissertation

We began this dissertation by discussing some common misconceptions about the literacy rates of the deaf and about the status of ASL as a natural language distinct from English – both of which delayed natural language processing researchers from exploring the problem of English-to-ASL machine translation. We then described the small number of research projects that have attempted to produce an English-to-ASL MT system. We discussed how these projects made only limited progress, and they did so by restricting their output to subsets of ASL phenomena – thus avoiding some important linguistic and animation issues. For example, these systems omitted phenomena depicting 3D information in the signing space (such as classifier predicates), did not specify some non-manual elements of the performance, and over-synchronized some parallel elements of the performance to the string of manual signs being performed.

The thesis of this dissertation is that previous attempts at developing ASL generation or MT systems have not succeeded because the design of these systems has been based too closely on NLP technologies designed for written languages. The restrictions these systems placed on their output enabled them to implement the ASL generation process using string-based NLP techniques and without using any models of spatial information, but the trade-off was that these restrictions also limited the quality of ASL output that could be produced. In order to build an ASL MT or generation system that produces a more realistic breadth of ASL output (without these restrictions), we should reconsider the design of the system with the requirements of sign language in mind (instead of attempting to re-cast the problem as written-language generation). The system's components should be able to access models that contain spatial information and be able to create a representation of the ASL signal that specifies the performance of the articulators of the signer's body (and their coordination relationships over time).

In support of this thesis, this dissertation has focused on how to generate ASL classifier predicates for an English-to-ASL MT system. We discussed how classifier predicate generation is an essential part of English-to-ASL MT; however, previous MT systems have been unable to address these phenomena. This dissertation has demonstrated how the development of new technologies that make use of linguistic models and surface-form representations specifically designed for ASL has enabled us to create a classifier predicate generator that can be used within an English-to-ASL MT system. Some of these technologies included: a visualization model of the 3D scene, a model of how discourse entities are associated with the signing space, a planning-based classifier predicate template formalism, a multichannel representation of ASL temporal structure, and a model of ASL articulators based on a small number of spatial data types and transformation functions. We have demonstrated that the design of our generator is ***well-specified*** and ***coherent*** (page 212), and that the generator is ***useful for MT*** (page 109), ***functional*** (page 261), ***scalable*** (page 212), and ***robust*** (page 109).

Major Technical Contributions

This dissertation has discussed several new computational linguistic representations and processing approaches tailored to the needs of ASL. This section will review the major technical contributions of this dissertation, and it will conclude with a summary of some important "firsts" of this project.

Modeling the Use of Space during an ASL Conversation

To produce classifier predicates and other ASL expressions that associate locations in space around a signer with entities under discussion, an English-to-ASL system must model what objects are being discussed in an English text, and it must map placeholders for these objects to locations in space around the signer's body. The ASL system described in this dissertation – and in earlier publications (Huenerfauth 2004b, 2005c) – has an explicit 3D model of how these placeholders representing discourse entities are positioned in the space around the signing character's body. For the classifier predicate generator, these placeholders are called “Ghosts,” and they indicate the layout of objects in a 3D scene being discussed.

Scene visualization software can be used to produce a 3D model of the arrangement and movement of objects discussed in an English input text, and this model is “mapped” onto a volume of space in front of the signer’s torso. This model of the signing space is used to guide the motion of the ASL signer’s hands during the performance of classifier predicates describing the motion of these objects.

Generating Classifier Predicates using Template-Based Planning

Given the 3D model of the “Ghosts” around the signer’s torso, the system uses a planning-based approach to determine how to move the signer’s hands, head-tilt, and eye-gaze to produce an animation of a classifier predicate (Huenerfauth 2004d, 2004b). The system stores a library of templates representing the various kinds of classifier predicates it may produce. These templates are implemented as hierarchical planning operators (with logical pre-conditions, sub-actions, and effects), allowing the system to decompose a classifier predicate into sub-components or to trigger other elements of performance that may be required during a grammatically correct ASL performance. This planning process also accesses a discourse model, which includes features relevant to classifier predicate generation. The planner builds an ASL surface-form representation (discussed below) that contains values for the articulators of the signer’s body.

Spatial Data Types Specified by Transformation Functions

The ASL linguistic models used inside the classifier predicate generator are fundamentally based upon a small number of spatial/linguistic data types whose values change over time and can be stored inside an ASL surface-form representation. This approach leads to an object-oriented software implementation: If we conceptualize the linguistic objects as feature bundles, then they can be implemented as classes that each have a set of members – the set of linguistic features each should store. The models which store these linguistic objects are merely containers (lists), and they can be implemented using object-oriented class templates.

Instead of storing static versions of values for locations, orientations, and other features inside of our models, we instead store dynamic values which record how the value changes during each time-unit of the performance. The values for the Articulators of the signer's body are distributed throughout a Partition/Constitute tree structure (discussed below) that represents the performance. The use of spatial transformation functions in this system allows us to isolate the planner from any 3D animation data; so, it can perform its classifier predicate planning on a symbolic representation.

Multi-Pathway Machine Translation Architecture

Because different forms of ASL signing (classifier predicates vs. lexical signing) may require different generation processes with different resources, we have argued that an English-to-ASL MT system should be implemented as a multi-path translation architecture. In addition to reflecting the variation in signing performance sometimes used by human ASL interpreters, this approach provides for a more graceful degradation of behavior for the system when it encounters difficult-to-translate inputs. This architecture takes the traditional MT pyramid and uses it in a different way: a single MT system incorporates several pathways at different heights of the pyramid (Huenerfauth 2004a). There is a direct, transfer, and interlingua (or at least a near-interlingua) classifier predicate pathway.

This novel machine translation architecture would allow a system to process certain input sentences (that require complex processing or which are important for some application) using a resource-intensive translation component and to process other inputs using a broader-coverage easier-to-build component. While the non-classifier-predicate portions of this design were not implemented for this dissertation, they were important to specify in order to demonstrate how a classifier predicate generator could be used within a complete English-to-ASL system.

Multi-Channel Representation of Linguistic Structure

While strings and syntax trees are used to represent written languages inside of NLP software, these encodings are difficult to adapt to a sign language. ASL lacks a standard writing system, and the multichannel nature of an ASL performance makes it difficult to encode in a linear single-channel string. This project developed a new formalism – the "Partition/Constitute Formalism" – for representing a linguistic signal in a multi-channel manner and for encoding temporal coordination and non-coordination relationships between portions of the signal (Huenerfauth 2005b, 2005d, 2006).

An Evaluation Methodology for ASL Machine Translation

ASL's multi-channel nature and lack of a writing system make it difficult to evaluate using standard automatic NLP metrics, which typically compare a string produced by a system to some human-produced “gold-standard” output string. We have therefore designed and conducted a user-based evaluation of our classifier predicate generator – native ASL signers evaluated three kinds of animations: (1) ASL output produced by the classifier predicate generator, (2) a Signed English animation, and (3) an animation of a 3D character whose movements are derived from those of a human wearing a motion-capture body suit and gloves. By conducting this study, we demonstrated the grammatical correctness, understandability, and naturalness of the animations produced by the classifier predicate generator; determined ways in which the animation output should be improved in future work; and learned how we can further improve the design of future ASL evaluation studies.

Important “Firsts” of this Project

There have been several accomplishments of this dissertation project that are important “firsts” for ASL computational linguistic research:

- This is the first project to implement a classifier predicate generator – it is the first time that a natural language processing system has generated this form of ASL animation. This classifier predicate generator design synthesizes and formalizes modern linguistic theories of ASL classifier predicates, and it uses a novel multichannel representation of the language signal being produced.

- This is the first ASL generation system that explicitly models the 3D arrangement of objects in a scene being discussed.[63] The system maps these objects on the signing space, and it uses this information during classifier predicate generation. To plan the animation performance without having to directly process this 3D data during most of the generation process, this project uses a novel "spatial transformation function" approach to specify the output values.

- This project has produced the first English-to-ASL MT design that includes a classifier predicate generator. The design shows how to incorporate a classifier predicate generation module with generation approaches for the non-classifier-predicate portions of ASL output in a multi-pathway design.

- This project conducted the first formal user-based evaluation of an ASL generation system – native ASL signers evaluated the output of the generator and the results were compared to animations of Signed English and motion-captured ASL. The evaluation suggested directions for future development of the generator, and the evaluation scores could be used to track the progress of a broader coverage system. This ASL evaluation design was novel, and the results of this study have suggested ways to refine the design in future work.

[63] Some researchers have studied ways of modeling the use of space around a signer during linguistic analysis of human-produced sign language performance (Liddell, 2003a) or have developed tools for computational representation of the signing space for such performances (Lenseigne and Dalle, 2005).

ASL Motivates New NLP Technologies

The design of the ASL generator described in this dissertation is quite different architecturally from traditional NLG systems for written languages. For instance, 3D graphics information is stored inside the system and the output is a multichannel representation of the changing values of a set of articulator values. These elements of the system have been motivated by linguistic properties of ASL:

- Since the classifier predicate generator was a focus of this project, designing an approach for producing the 3D locations and orientations in a classifier predicate animation was an important goal. This has been addressed by the system's use of a 3D model produced by scene-visualization software, a planning-based classifier predicate generator (whose planning operators are templates that are parameterized on the spatial values of objects in the scene), and a set of spatial data types whose values can be specified by transformation functions.
- The multichannel nature of ASL has led this project to study mechanisms for coordinating the values of the articulators used during generation. The development of the Partition/Constitute timing formalism and a set of linguistic/spatial values that can be stored inside this representation have allowed this system to represent the coordination and non-coordination relationships in the ASL signal (and use this information to control an animation output).
- While we have focused on the classifier predicate subsystem of the language, we've also tried to make design choices with the Lexical Signing subsystem of the language in mind. This has motivated: a multi-path MT architecture, a

discourse model capable of storing data relevant to both subsystems, models of the space around the signer capable of storing placeholders for either type of signing, and an ASL surface-form representation suitable for either subsystem.

Applications of These Technologies beyond ASL

While the technologies outlined above have been developed for ASL, they also have applications for the creation of natural language processing software for spoken/written languages. These technologies could be applied to the generation of multimodal language signals, generation with multiple pathways, translation of spatially descriptive text, or the generation/evaluation of gesture for embodied conversational agents.

Multimodal Multichannel Generation

ASL generation required a linguistic representation that coordinated multiple channels of linguistic output, but it's not unique in this requirement. In fact, generation of a communication signal for any language may require these capabilities (even for spoken languages like English). The discussion of the P/C Formalism in Chapter 6 has already suggested that researchers studying the generation of meaningful gestures for Embodied Conversational Agents (Kopp et al., 2004) could also benefit from this form of representation (and the accompanying design of generation algorithms which produce this multichannel representation). If these researchers wish to produce gestures that are more linguistically conventional or internally complex, then a P/C-based linguistic representation would provide advantages.

Other computational linguistic applications could benefit from an NLG design with multiple linguistic channels (and indirectly benefit from the development of ASL NLG technology). For instance, NLG systems producing speech output could encode prosody, timing, volume, intonation, or other vocal data as multiple linguistically-determined channels of the output (in addition to a channel for the string of words being generated). Therefore ASL NLG research not only has benefits for deaf users, but it also serves as a research vehicle for NLG technology to produce a variety of richer-than-text linguistic communication signals.

Multi-Systemic Generation of Written Languages

The overall English-to-ASL MT design of this system included multiple ASL generation pathways inside of the single system. This capability would allow the design to produce linguistic output in several different subsystems of the language. For ASL, some sentences could use the planning-based pathway to produce classifier predicate output while others could use a different pathway to produce lexical signing output.

This approach to building an MT system can also be used to blend MT methods in a written-language MT system. This multi-path architecture can combine a resource-intensive deep-processing MT method for difficult (or important) inputs and a resource-light broad-coverage MT method for other inputs. Merging multiple MT approaches in one system alleviates the trade-off between divergence-handling power and domain specificity, thus making resource-intensive approaches (e.g. interlingua or deep-transfer) practical for applications that require broad linguistic coverage. This architecture is useful when a system must translate a variety of texts but perform deeper processing on texts within particular important or complex domains. While ASL possessed two quite distinct subsystems, the desired output of any language might contain a sublanguage of text that the MT design wishes to generate using a distinct processing approach. The designer may wish to develop resource-intensive (transfer or interlingua) MT approaches for the sublanguage of text that is of interest and resource-lighter (broader coverage) MT approaches could handle the of the target language output.

While this particular English-to-ASL MT system did not contain any statistical pathways, nothing would prevent their use in a multi-path MT architecture. For instance, statistical approaches could be used to develop a direct pathway, and hand-built analysis and transfer rules could be used to build a transfer pathway for a subset of the inputs. A developer could thus use a stochastic approach for most inputs but manually override the MT process for certain texts (that are important or whose translation is well understood). Likewise, a transfer pathway could be developed that contains statistically-induced analysis or transfer rules, and an interlingua MT pathway could be built manually for specific domains of interest.

Interlinguas for Spatially Descriptive Text

The 3D scene visualizations in this ASL generation system serve as an intermediary between the English and ASL halves of a translation process. Chapter 5 discussed how the spatial 3D model is language-neutral and thus acts like an interlingua for the sub-domain of texts that describe 3D motion. (We actually discussed how the use of links to English predicate argument structures inside of the classifier predicate templates makes our MT design more of a deep-transfer or near-interlingua based approach.) In the same way that the 3D Visualization Scene aids the generation of ASL spatial expressions (in the form of classifier predicates), a spatially-rich 3D-model interlingua could be used in a written-language MT systems to generate spatially descriptive text.

Generation of Gestures for Embodied Conversational Agents

The way in which this generator produces the complex body movements of classifier predicates describing 3D spatial concepts suggests new ways to produce manual gestures for other Embodied Conversational Agents – especially gestures which convey spatial or iconic (visually representative) information. Unlike work by (Tepper et al, 2004) which generates iconic gestures from a logical-predicate-based representation of objects under discussion, this classifier predicate generator uses a 3D spatial model of these objects. This richer model (and the templated, planning-based generation approach used for classifier predicates) could be used to build a more robust gesture generation system that produces 3D-topologically-representative gestures.

Evaluation of Gestures for Embodied Conversational Agents

The design of the evaluation study conducted on the classifier predicate generator has applications for the evaluation of the animation output of gesture-generation systems. In particular, embodied conversational agent researchers could use upper-baseline animations of a language performance (with gestures) recorded from a human by motion-capture or via one of the alternate approaches discussed in the previous chapter. This could serve as an important point of comparison when evaluating the animation output of their embodied conversation agents.

As we saw in our ASL study, respondents' perception of how well they understood an animation did not correspond very tightly with their actual success at the matching-task. If the same is true of animations of an embodied conversation agent that is performing gestures to help convey spatial information, then it could be useful for evaluation studies of these animations to also include a quantitative measure of comprehension like a matching-task.

How ASL NLP Research Contributes to the Field

We've seen that new technologies developed in this dissertation have applications for the generation of written language, multimodal linguistic signals, and embodied conversational agents. While it was this project's focus on ASL that motivated us to develop these approaches, they are useful for languages that are more commonly studied by natural language processing researchers. One reason why these techniques had not been developed previously is that it was the special set of priorities and requirements of ASL generation that helped to clarify the need for them.

This highlights an important contribution that research into ASL generation and machine translation can have for the field of natural language processing. As a language that lacks a standard written form, that incorporates 3D information into its performance, that has multiple subsystems, and that is naturally encoded as multiple parallel channels, ASL presents a unique set of challenges. This pushes the boundaries of current NLP technology, and it motivates researchers to develop new techniques – many of which have applications to other unexpected areas of NLP research.

Bibliography

H. Adachi, S. Yosizawa, M. Fujita, T. Matsumoto, & K. Kamata. 2001. Analysis of News Sentences with Sign Language and Sign Translation Processing. IPSJ SIGNotes Natural Language Abstract No.091 - 003.

Ascension Technologies. 2006. ReActor©2: Digital Active-Optical Mocap System. Retrieved July 19, 2006, from Ascension Technologies website: http://www.ascension-tech.com/products/reactor.php

N.I. Badler, J. Allbeck, S.J. Lee, R.J. Rabbitz, T.T. Broderick, and K.M. Mulkern. 2005. New Behavioral Paradigms for Virtual Human Models. SAE International Digital Human Modeling for Design and Engineering.

N. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, S. Lee, H. Shin, & M. Palmer. 2000. Parameterized action representation and natural language instructions for dynamic behavior modification of embodied agents. AAAI Spring Symposium.

A. Bahill, D. Aandler, and L. Stark. 1975. Most naturally occuring human saccades have magnitudes of 15 deg or less. In Investigative Ophthalmol., pp. 468–469.

C. Baker. 1977. Regulators and Turn-Taking in American Sign Language Discourse. In L.A. Friedman (Ed.), On the Other Hand: New Perspectives on American Sign Language. New York: Academic Press, pp. 215-236.

W. Baldwin & G. Strawn. 1991. Multidimensional Trees. Theoretical Computer Science 84:2, pp. 293-311.

J. A. Bangham, S. J. Cox, R. Elliot, J. R. W. Glauert, I. Marshall, S. Rankov, & M. Wells. 2000. Virtual signing: Capture, animation, storage and transmission – An overview of the ViSiCAST project. IEEE Seminar on Speech and language processing for disabled and elderly people.

R. Bindiganavale, W. Schuler, J. Allbeck, N. Badler, A. Joshi, & M. Palmer. 2000. Dynamically Altering Agent Behaviors Using Natural Language Instructions. 4th International Conference on Autonomous Agents.

S. Bird & M. Lieberman. 2000. A Formal Framework for Linguistic Annotation. Speech Communication 33:1,2, pp. 23-60.

D. Brentari. 1990. Theoretical Foundations of American Sign Language Phonology. Ph.D. dissertation, University of Chicago.

D. Brentari. 2005. The Prosodic Structure of Sign Language Classifiers: A cross-linguistic analysis . Linguistic Society of America, January 6-8, Chicago.

J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone. 1994. Animated Conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In ACM SIGGRAPH Annual Conference Series, pp. 413-420.

J. Cassell, M. Stone, & H. Yan. 2000. Coordination and Context-Dependence in the Generation of Embodied Conversation. International Natural Language Generation Conference, Mitzpe Ramon, Israel.

J. Cassell, H. Vilhjálmsson, & T. Bickmore. 2001. BEAT: the Behavior Expression Animation Toolkit. SIGGRAPH 2001, Los Angeles, CA, USA.

J. Coates and R. Sutton-Spence. 2001. Turn-Taking Patterns in Deaf Conversation. Journal of Sociolinguistics, Volume 5, Number 4, November 2001, pp. 507-529(23).

G. Coulter, (ed.). 1993. Current issues in ASL phonology. Phonetics and Phonology, Volume 3. New York, San Francisco, London: Academic Press.

R. Coyne & R. Sproat. 2001. WordsEye: an automatic text-to-scene conversion system. SIGGRAPH 2001, Los Angeles, CA, USA. pp. 487-496.

M.J. Davidson, K. Alkoby, E. Sedgwick, A. Berthiaume, R. Carter, J. Christopher, B. Craft, J. Furst, D. Hinkle, B. Konie, G. Lancaster, S. Luecking, A. Morris, J. McDonald, N. Tomuro, J. Toro, & R. Wolfe. 2000. Usability Testing of Computer Animation of Fingerspelling for American Sign Language. DePaul CTI Research Conference, Chicago, IL, November 4, 2000.

M.J. Davidson, K. Alkoby, E. Sedgwick, R. Carter, J. Christopher, B. Craft, J. Furst, D. Hinkle, B. Konie, G. Lancaster, S. Luecking, A. Morris, J. McDonald, N. Tomuro, J. Toro, & R. Wolfe. 2001. Improved Hand Animation for American Sign Language. Technology And Persons With Disabilities Conference.

A. DeMatteo. 1977. Visual Analogy and the Visual Analogues in American Sign Language. In Lynn Friedman (ed.), On the Other Hand: New Perspectives on American Sign Language. pp 109-136. New York: Academic Press.

B. Dorr. 1992. The Use of Lexical Semantics in Interlingual Machine Translation. Machine Translation, 7:3, pp. 135-193.

B. Dorr, P. Jordan, & J. Benoit. 1998. A Survey of Current Paradigms in Machine Translation. Technical Report: LAMP-TR-027/UMIACS-TR-98-72/CS-TR-3961, University of Maryland, College Park, December 1998.

B. Dorr & C. Voss. 2000. A Multi-Level Approach to Interlingual MT: Defining the Interface between Representational Languages. In L. Iwanska & S. Shapiro (eds.), Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language. AAAI Press/MIT Press. pp. 207-248.

P. Eccarius and D. Brentari. (In press). A Cross-linguistic Study of Two-Handed, Sign Languages Classifiers: Phonological and prosodic restrictions. Lingua.

R. Elliott, J. Glauert, V. Jennings, & J. Kennaway. 2004. An Overview of the SiGML Notation and SiGML Signing Software System. Workshop on the Representation and Processing of Signed Languages, 4th International Conference on Language Resources and Evaluation.

K. Emmorey, (ed.). 2003. Perspectives on Classifier Constructions in Sign Languages. Workshop on Classifier Constructions, La Jolla, San Diego, CA.

N. Frishberg. 1975. Arbitrariness and Iconicity: Historical change in American Sign Language. Language, 51, 696-719.

J. Furst, K. Alkoby, A. Berthiaume, P. Chomwong, M.J. Davidson, B. Konie, G. Lancaster, S. Lytinen, J. McDonald, L. Roychoudhuri, J. Toro, N. Tomuro, & R. Wolfe. 2000. Database Design for American Sign Language. In Proceedings of the ISCA 15th International Conference on Computers and Their Applications (CATA-2000). pp. 427-430.

A.B. Grieve-Smith. 2002. English to American Sign Language Machine Translation of Weather Reports. In D. Nordquist, (ed.), Proceedings of the Second High Desert Student Conference in Linguistics. High Desert Linguistics Society, Albuquerque, NM.

E. Gu and N. Badler. 2006. Visual Attention and Eye Gaze during Multiparty Conversations with Distractions, Intelligent Virtual Agent.

J. Holt. 1991. Demographic, Stanford Achievement Test - 8th Edition for Deaf and Hard of Hearing Students: Reading Comprehension Subgroup Results.

M. Huenerfauth. 2003. Survey and Critique of ASL Natural Language Generation and Machine Translation Systems. Technical Report MS-CIS-03-32, Computer and Information Science, University of Pennsylvania.

M. Huenerfauth. 2004a. A Multi-Path Architecture for Machine Translation of English Text into ASL Animation. HLT-NAACL Student Workshop.

M. Huenerfauth. 2004b. Spatial Representation of Classifier Predicates for Machine Translation into American Sign Language. Workshop on Representation and Processing of Signed Languages, LREC 2004.

M. Huenerfauth. 2004c. American Sign Language Natural Language Generation and Machine Translation. The 6th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS 2004), Doctoral Consortium Presentation and Poster Session. Atlanta, Georgia, USA.

Huenerfauth, M. 2004d. Spatial and Planning Models of ASL Classifier Predicates for Machine Translation. 10th International Conference on Theoretical and Methodological Issues in Machine Translation: TMI 2004, Baltimore, MD, USA.

M. Huenerfauth. 2005a. American Sign Language Natural Language Generation and Machine Translation. ACM SIGACCESS Accessibility and Computing. New York: ACM Press. Issue 81 (January 2005).

M. Huenerfauth. 2005b. American Sign Language Generation: Multimodal NLG with Multiple Linguistic Channels. Student Research Workshop, The 43rd Annual Meeting of the Association for Computational Linguistics. Ann Arbor, MI, USA.

M. Huenerfauth. 2005c. American Sign Language Spatial Representations for an Accessible User-Interface. 3rd International Conference on Universal Access in Human-Computer Interaction. Las Vegas, NV, USA.

M. Huenerfauth. 2005d. Representing Coordination and Non-coordination in an American Sign Language Animation. The 7th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS 2005), Baltimore, MD, USA.

M. Huenerfauth. 2006. Representing Coordination and Non-Coordination in American Sign Language Animations. Behaviour & Information Technology, Volume 25, Issue 4, July 2006, pp. 285-295.

Immersion Corp. 2006. CyberGlove® II Wireless Data Glove. Retrieved July 19, 2006, from Immersion Corporation website: http://www.immersion.com/3d/products/cyber_glove.php

J. R. Kennaway. 2001. Synthetic animation of deaf signing gestures. In I. Wachsmuth & T. Sowa, (eds.), Proceedings of the 4th International Workshop on Gesture and Sign Language Based Human-Computer Interaction, Lecture Notes in Artificial Intelligence, Volume 2298.

K. Kipper, B. Snyder, & M. Palmer. 2004. Extending a Verb-lexicon Using a Semantically Annotated Corpus. In the Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004).

S. Kopp, P. Tepper, & J. Cassell. 2004. Towards Integrated Microplanning of Language and Iconic Gesture for Multimodal Output. International Conference on Multimodal Interfaces, State College, PA, USA.

H. Lane, R. Hoffmeister, & B. Bahan. 1996. A Journey into the Deaf World. San Diego, CA: DawnSignPress, p. 42.

S.P. Lee, J. Badler, N. Badler. 2002. Eyes Alive. Proceedings of ACM SIGGRAPH.

B. Lenseigne and P. Dalle. 2005. A Tool for Sign Language Analysis through Signing Space Representation. Sign Language Linguistics and the Application of Information Technology to Sign Languages, Milan, Italy, June 2005.

S. Liddell. 1977. An Investigation into the Syntactic Structure of ASL. Ph.D. dissertation, University of California, San Diego.

S. Liddell & R. Johnson. 1989. American Sign Language: The Phonological Base. Sign Language Studies, Volume 64, pp. 195-277.

S. Liddell. 2003a. Grammar, Gesture, and Meaning in American Sign Language. UK: Cambridge University Press.

S. Liddell. 2003b. Sources of Meaning in ASL Classifier Predicates. In Karen Emmorey (ed.), Perspectives on Classifier Constructions in Sign Languages. Workshop on Classifier Constructions, La Jolla, San Diego, California.

Y. Liu. 2003. Interactive Reach Planning for Animated Characters Using Hardware Acceleration. Doctoral Dissertation, Computer and Information Science, University of Pennsylvania.

D. Lonsdale, A. Franz, & J. Leavitt. 1994. Large-scale Machine Translation: An Interlingua Approach. In Proceedings of IEA/AIE-94.

S.I. Lu, H. Matsuo & Y. Nagashima. 1997. Towards a Dialogue System Based on Recognition and Synthesis of Japan Sign Language. Gesture and Sign Language in Human-Computer Interaction.

M. Mandel. 1977. Iconic devices in American Sign Language. In: L.A. Friedman (Ed.), On the Other Hand: New perspectives on American Sign Language. New York: Academic Press. pp. 57-107

K. Marriot & B. Meyer. 1996. Towards a Hierarchy of Visual Languages. In the Proceedings of the AVI'96 Workshop on the Theory of Visual Languages and Computing, Volume 2, pp. 311-331.

C. Martell. 2002. FORM: An extensible, kinematically-based gesture annotation scheme. In the Proceedings of the 3rd International Conference on Language Resources and Evaluation.

I. Marshall & É. Sáfár. 2001. Extraction of semantic representations from syntactic SMU link grammar linkages. In G. Angelova, (ed.), Proceedings of Recent Advances in Natural Language Processing (RANLP), pp. 154-159, Tzigov Chark, Bulgaria, September.

D. MacLaughlin. 1997. The Structure of Determiner Phrases: Evidence from American Sign Language. Ph.D. dissertation, Boston University, Boston, MA.

A. Mital and H.F. Faard. 1990. Effects of Sitting and Standing, Reach Distance, and Arm Orientation on Isokinetic Pull Strengths in the Horizontal Plane. International Journal of Industrial Ergonomics, 6. pp. 241-248.

R. Mitchell. 2004. How many deaf people are there in the United States. Retrieved June 28, 2004, from Gallaudet Research Institute, Graduate School and Professional Programs, Gallaudet University Web site: http://gri.gallaudet.edu/Demographics/deaf-US.php

J. Morford & J. MacFarlane. 2003. Frequency Characteristics of American Sign Language. Sign Language Studies, 3:2.

D. Nau, H. Muñoz-Avila, Y. Cao, A. Lotem, and S. Mitchell. 2001. Total-Order Planning with Partially Ordered Subtasks. In Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI-2001), Seattle, WA, USA.

D. Nau, T.-C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman. 2003. SHOP2: An HTN Planning System. Journal of Artificial Intelligence Research, Volume 20, pp. 379-404.

C. Neidle, D. Kegl, D. MacLaughlin, B. Bahan, & R.G. Lee. 2000. The Syntax of American Sign Language: Functional Categories and Hierarchical Structure. Cambridge, MA: The MIT Press.

C. Neidle, S. Sclaroff, & V. Athitsos. 2001. SignStream™: A Tool for Linguistic and Computer Vision Research on Visual-Gestural Language Data. In Behavior Research Methods, Instruments, and Computers 33:3, pp. 311-320.

D. Newkirk. 1987. SignFont Handbook. San Diego: Emerson and Associates.

M. Ohki, H. Sagawa, T. Sakiyama, E. Oohira, H. Ikeda, & H. Fujisawa. 1994. Pattern recognition and synthesis for sign language translation system. In Proceedings of the First International ACM Conference on Assistive Technologies. ACM Press.

C. Padden. 1988. Interaction of Morphology and Syntax in American Sign Language. Outstanding Dissertations in Linguistics, Series IV. New York: Garland Press.

C. Padden and D. Perlmutter. 1987. American Sign Language and the Architecture of Phonological Theory. Natural Language and Linguistic Theory, 5. pp 335-375.

A. Pandya and A.J. Micocci. 1989. Dynamic Strength Data on the Shoulder, Elbow and Wrist. NASA Johnson Space Center.

S. Pasquariello and C. Pelachaud. 2001. Greta: A simple facial animation engine. In 6th Online World Conference on Soft Computing in Industrial Applications.

É. Sáfár & I. Marshall. 2001. The architecture of an English-text-to-Sign-Languages translation system. In G. Angelova, (ed.), Proceedings of Recent Advances in Natural Language Processing (RANLP), pp. 223-228, Tzigov Chark, Bulgaria.

É. Sáfár & I. Marshall. 2002. Sign language translation via DRT and HPSG. In A. Gelbukh (ed.), Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, CICLing, Mexico, Lecture Notes in Computer Science 2276, pp. 58-68, Springer Verlag.

B. S. Schick. 1987. The Acquisition of Classifier Predicates in American Sign Language. Ph.D. Dissertation. Purdue University.

W. Schuler. 2003. Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03), Sapporo, Japan.

K. Shoemake. 1985. Animating Rotation with Quaternion Curves. Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press. pp. 245-254.

d'A.L. Speers. 2001. Representation of American Sign Language for Machine Translation. PhD Dissertation, Department of Linguistics, Georgetown University.

T. Supalla. 1978. Morphology of Verbs of Motion and Location. In F. Caccamise & D. Hicks (eds.), Proceedings of the Second National Symposium on Sign Language Research and Teaching, pp. 27-45. Silver Spring, MD: National Association for the Deaf.

T. Supalla. 1982. Structure and Acquisition of Verbs of Motion and Location in American Sign Language. Ph.D. Dissertation, University of California, San Diego.

T. Supalla. 1986. The Classifier System in American Sign Language. In C. Craig (ed.), Noun Phrases and Categorization, Typological Studies in Language, Volume 7. pp. 181-214. Philadelphia: John Benjamins.

N. Suszczańska, P. Szmal, & J. Francik. 2002. Translating Polish Texts into Sign Language in the TGT System. 20th IASTED International Multi-Conference. Applied Informatics AI 2002, pp. 282-287, Innsbruck, Austria.

V. Sutton. 1998. The Signwriting Literacy Project. Impact of Deafness on Cognition AERA Conference, San Diego, CA.

L. Talmy. 2003. The Representation of Spatial Structure in Spoken and Signed Language. In Emmorey, K. (ed.), Perspectives on Classifier Constructions in Sign Languages, Mahwah, NJ: Lawrence Erlbaum.

P. Tepper, S. Kopp, & J. Cassell. 2004. Content in Context: Generating Language and Iconic Gesture without a Gestionary. Workshop on Balanced Perception and Action in ECAs, Autonomous Agents & Multiagent Systems (AAMAS-04). New York, NY.

M. Tokuda & M. Okumara. 1998. Towards Automatic Translation from Japanese into Japanese Sign Language. Assistive Technology and AI, LNAI 1458, pp. 97-108, Springer Verlag.

M. Tucci, G. Vitiello, & G. Costagliola. 1994. Parsing Nonlinear Languages. In IEEE Transactions in Software Engineering, 20:9.

C. Valli & C. Lucas. 2000. Linguistics of American Sign Language, 3rd Edition, Washington, DC: Gallaudet University Press.

L. van Zijl & D. Barker. 2003. A Machine Translation System for South African Sign Language. In Proceedings of Afrigraph 2003, Cape Town, South Africa.

T. Veale, A. Conway, & B. Collins. 1998. The challenges of cross-modal translation: English to sign language translation in the ZARDOZ system. In Machine Translation, Volume 13, pp. 81-106.

M. Van Herreweghe. 2002. Turn-Taking Mechanisms and Active Participation in Meetings with Deaf and Hearing Participants in Flanders. In C. Lucas (Ed.), Turn-

Taking, Fingerspelling, and Contact in Signed Languages. The Eighth Volume in the Sociolinguistics in Deaf Communities Series.

M. Verlinden, C. Tijsseling & H. Frowein. 2001. A Signing Avatar on the WWW. In K. Nordby, (ed.), Proceedings of the 18th International Symposium on Human Factors in Telecommunication, Bergen, Norway.

C. J. Wideman & E. M. Sims. 1998. Signing Avatars. Technology and Persons with Disabilities Conference.

R. Wolfe, K. Alkoby, J. Barnett, P. Chomwong, J. Furst, G. Honda, G. Lancaster, F. Lavoie, S. Lytinen, J. McDonald, L. Roychoudhuri, C. Taylor, N. Tomuro, & J. Toro. 1999. An Interface for Transcribing American Sign Language.

L. Xu & W. Gao. 2000. Study on translating Chinese into Chinese Sign Language. Journal of Computer Science and Technology, 15:5. pp. 485-490.

L. Zhao, K. Kipper, W. Schuler, C. Vogler, N. Badler, & M. Palmer. 2000. A Machine Translation System from English to American Sign Language. Association for Machine Translation in the Americas.

L. Zhao, Y. Liu, N.I. Badler. 2005. Applying Empirical Data on Upper Torso Movement to Real-time Collision-free Reach Tasks. SAE Digital Human Modeling Conference, Iowa City, IA.