The AAAI 2017 Spring Symposium on
Computational Construction Grammar and Natural Language Understanding
Technical Report SS-17-02

Cognitive Modeling Approaches to Language Comprehension Using Construction Grammar

Peter Lindes and John E. Laird
University of Michigan
{plindes, laird}@umich.org

Abstract

This paper examines the relationship between modeling human sentence comprehension using cognitive architectures and approaches to linguistic knowledge representation using construction grammars. We review multiple computational models of language understanding that vary in their use of construction grammar and cognitive architectures. We present a case study: Lucia, that uses Embodied Construction Grammar (ECG) within the Soar cognitive architecture to comprehend language used to instruct an embodied agent. We also examine the tradeoffs between alternative approaches to representing and accessing linguistic knowledge within a cognitive architecture and suggest future research.

Introduction

How do human beings comprehend natural language? Many disciplines within cognitive science have contributed knowledge to help answer this question. Psychologists and psycholinguists have gathered empirical data on human eye movements, reaction times, reading times, priming effects, and so on related to language comprehension. Cognitive linguists have developed theories of image schemas, metaphor, lexical semantics, and construction grammar to gain insight into the structure of language. Research in cognitive architecture has provided insight on the computational building blocks underlying language processing.

We examine language comprehension using construction grammars embedded within cognitive models – models that attempt to model the human comprehension process, and more specifically, models implemented within a cognitive architecture. Cognitive architectures define the fixed structures and functions underlying cognition, including the representation, storage, and access of knowledge in both short-term and long-term memory, decision making, and interfaces to perception and action. Cognitive architectures embody the hypothesis articulated by Allen Newell (1990) that the same computational structures are used across all human mental activity, including language processing. We consider models of language comprehension in both the Soar and ACT-R cognitive architectures.

Within the context of cognitive architecture, the question of how do humans comprehend natural language leads us to the following questions: How is knowledge about language represented? How is that knowledge encoded in and retrieved from the different memories of a cognitive architecture? How is knowledge used to dynamically create actionable meaning representations from input sentences rife with ambiguities and ungrammaticalities?

In this paper, we explore seven different sentence processing systems. These systems vary along many dimensions: how they structure their knowledge; whether they attempt to model human behavior; whether they focus only on syntactic processing or also include semantic processing that creates structures that are used for producing action; which memories they use to represent the different types of knowledge used in language processing; and how they process that knowledge, especially to deal with ambiguity.

Models of Linguistic Knowledge

Fundamental to models of sentence processing is the representation and structure of linguistic knowledge. Construction grammars (Hoffman and Trousdale, 2013) are a set of theories as to how linguistic knowledge is structured and represented. These theories posit the idea of a *construction*, which is a unit of lexical or grammatical knowledge that pairs a form with a corresponding meaning. Many variations of construction grammar formalisms have been developed (Goldberg 2013). The two frameworks that are most relevant here are Embodied Construction Grammar (ECG; Bergen and Chang 2013) and Fluid Construction Grammar (FCG; Steels 2013).

Embodied Construction Grammar (ECG)

ECG (Eppe et al. 2016; Feldman, Dodge, and Bryant 2009) is rooted in semantic representations based on theories of image schemas. It relates linguistic forms to these schemas which can represent perception and mental models, and also to action schemas, which describe physical actions that can be performed by an agent such that the parameters of an action can be filled in by information from the language input. Thus the grammatical representation is closely tied to the perception and action of an embodied agent.

Bryant (2008) created a parser using ECG as its grammar definition language and a probabilistic, left-corner parsing algorithm which searches for a best fit representation of a sentence using the given grammar. This parser has been used in a number of applications of ECG (Chang, 2008; Eppe et al. 2016). Each sentence produces a semantic representation called a *semspec*. Recent work uses this parser to provide a sentence comprehension capability to robots.

Fluid Construction Grammar (FCG)

FCG (Steels and Hild, 2012) is a construction grammar formalism that has been used with robots and for experiments in learning linguistic knowledge by an agent or group of agents. It has the feature that the same constructions can be used for both comprehension and language production.

The comprehension processor for FCG "supports standard heuristic best-first search" (Steels, De Beule, and Wellens 2012). One of the major thrusts of FCG is that the parser has a number of facilities to enhance its robustness (Steels and van Trijp 2011) for missing or incomplete items. It can coerce items to change their properties, and learning processes can deal with unknown words and meanings.

Analysis of Construction Grammar

A construction grammar provides a formalization for representing the connections between syntactic and semantic knowledge. Constructions for both lexical and grammatical constructions can include semantic information, which is a key point in construction grammar models of language (Goldberg 1995). As Jurafsky (1996) points out, construction grammars have an advantage over traditional context-free grammars in that an instance of a construction is an actual data structure, not just a label, and can thus hold semantic and grammatical feature data.Thus, constructions provide a declarative map from form to meaning. The cognitive models we discuss below provide a dynamic process to construct meanings from input utterances one word at a time.

Modeling Human Processing

Both ECG and FCG provide theories of how linguistic knowledge is represented and how input sentences are parsed to create semantic representations. Both have been applied to embodied robotic agents. However, they are not embodied in a cognitive architecture nor do they attempt to model human comprehension processes.

In contrast, there have been multiple parsers developed in cognitive architectures, specifically within ACT-R (Anderson et al. 2004) and Soar (Laird 2012; Newell 1990). Although ACT-R and Soar differ in many details, at an abstract level of analysis they share the same overall structure and processing cycle. They both have a basic processing cycle that is controlled by knowledge retrieved from procedural memory. This knowledge is encoded as rules that test the current situation in order to decide what action, either internal or external, should be performed. The current situation is encoded as symbolic structures in working memory (which are buffers in ACT-R). The contents of working memory come from perception and retrievals from long-term memories, which include the procedural memory as well as long-term declarative memories. The internal actions add, remove, or modify structures in working memory; or retrieve information from declarative memories. The external actions initiative motor actions. All of these processes are modulated by metadata that the architecture maintains about its memories and processes. For example, the retrieval of information from declarative memory can be biased using the frequency and recency of prior accesses of long-term memory structures (base-level activation), as well as associations between memory structures (spreading activation).

Because of the context-dependent retrieval of knowledge from procedural and declarative long-term memories, ACT-R and Soar do not have the same predetermined sequential execution structure that is ubiquitous in standard programming languages. Instead, each decision is determined by the current situation – the active goals, perception, and internal reasoning state of the system.

Soar and ACT-R have similar procedural learning mechanisms that incrementally create new productions based on the co-occurrence of production firings. Soar's mechanism, called chunking, creates productions that summarize the processing in subgoals, while ACT-R's mechanism, called production composition, composes two productions that fire in sequence into a single production.

In Soar, the processing consists of a series of decision cycles where a single operator is selected and applied. Any complex behavior, such as sentence processing, balancing a check book, or writing a symposium paper, is composed of sequences of primitive operators. When mapping to human behavior, Newell (1990) estimates that each decision cycle takes ~50 ms. Thus, since adult humans can read at a

rate of around 240 words/minute, with speech processing being somewhat slower, language processing should average ~4-5 decision cycles per word. This provides a tight constraint on models of sentence processing. We call timing measurements based on the number of decision cycles, assumed to take ~50 ms in humans, *simulated real time* measurements.

The need to keep up with the continual stream of new words leads the sentence processing systems we describe below to perform as much processing as possible on each word, including lexical access, syntactic, and semantic processing. Although there may be additional processing at clause and sentence boundaries, there is insufficient time for extensive processing of the complete sentence, so it is critical that as much syntactic and semantic processing as possible is done as individual words arrive. This leads to incremental, left-to-right, word-at-a-time, processing, where a data structure representing both the semantic and the syntactic structure of the input sentence is built up. In order to produce actionable meanings, and to assist in resolving ambiguities, semantic structures are grounded to the agent's perception and action capabilities as soon as possible (Just and Carpenter 1987). When mistakes are made, as they inevitably are in ambiguous sentences, the system attempts to make to make a local repair (Lewis 1993), or if that fails, just keep processing and extract whatever meaning it can. It is possible to attempt a complete reparse during reading, by starting over at the beginning of a clause or sentence.

Although the cognitive systems we discuss have many similarities (many later systems were inspired by earlier systems), they vary in how linguistic knowledge is structured, whether semantic processing is done, how linguistic knowledge is distributed over long-term memories, whether there are explicit predictions of future structures, and how ambiguities are handled.

NL-Soar

NL-Soar (Lehman, Lewis, and Newell 1991; Newell 1990) was the first attempt to apply a cognitive architecture to sentence processing. NL-Soar did integrated, incremental processing based on comprehension operators, each of which maps form to meaning for a single input word. Using the Soar features of impasses, subgoals, and chunking, these operators compiled different types of knowledge that initially required deliberative processing, into reactive processing that modeled adult performance. In NL-Soar, all temporary parsing data was stored in Soar's working memory.

The dynamic process of parsing in NL-Soar follows a single path rather than maintaining several alternative options. Expectations can be generated at each word so that processing for later words can combine both bottom-up and top-down knowledge. When an earlier choice later turns out to be incorrect, limited local repair can correct it. Lewis (1993) showed that this model can run in simulated real time and exhibits limitations similar to humans in dealing with garden path sentences and difficult center embeddings.

Grammatical knowledge in NL-Soar was stored in handwritten production rules in procedural memory. Early versions used a dependency grammar representation (Lehman, Lewis, and Newell 1991a), while Lewis (1993) changed this to use X-bar theory and Government and Binding. During processing, both a syntactic and a semantic model of an utterance were built.

Rubinoff and Lehman (1994) report on a version of NL-Soar which included language production as well as comprehension, and that interleaved this production with other task processing. They describe two systems built where NL-Soar was used to provide language comprehension and production. NTD-Soar simulated a NASA test director, and TacAir-Soar simulated the pilot of a military aircraft.

Lonsdale (2011) created a variation of NL-Soar, called XNL-Soar, that was designed to use the Minimalist Program theory of language. This system built semantic representations, but without any grounding in an external environment. A version of XNL-Soar processed Japanese (instead of English) with only a few changes to the grammar other than replacing the lexical items. The XNL-Soar system used an order of magnitude fewer production rules than NL-Soar because it stored much of its lexical and semantic knowledge in Soar's declarative semantic memory.

Lewis's ACT-R Model

Lewis and Vasishth (2005) developed a sentence processing model using the ACT-R cognitive architecture. This model uses many concepts derived in terms of word-by-word processing from NL-Soar, but its internal processing differs in how it accesses intermediate linguisitic structures as they are constructed. Lewis, Vasishth, and Van Dyke (2006) discuss the architectural principles underlying this model.

Central to the model is that ACT-R has a very limited working memory, with only a few items being available in buffers. Therefore, declarative memory is used in this model to store both lexical knowledge and the intermediate results of parsing. Knowledge to match grammatical constructions is encoded in production rules that are built by hand. The division of knowledge between declarative and procedural memories in this model is based in part on cognitive neuroscience models (Ullman 2004). Lexical knowledge is in declarative memory while all grammatical knowledge is stored procedurally.

The parsing process in this model is based on cue-based parsing theory (Van Dyke and Lewis 2003). As words are

processed, the feature bundles representing the words and expectations for larger grammatical structures are stored in declarative memory. Then later words can encode cues that are used to search this memory for the previous expectations. For example, in the sentence *Melissa knew that the toy from her uncle in Bogotá arrived today.* when the verb *arrived* is processed, a cue is created that searches declarative memory for the expected sentence structure built for *the toy* as the subject but not yet having a verb. When this structure is found, the verb is merged in. This model focuses on syntax and does not produce a semantic representation. Its linguistic knowledge is, like later versions of NL-Soar, based on X-bar theory and Government and Binding.

This model takes advantage of the activation mechanisms in ACT-R to give precise predictions of processing times that are then compared to empirical measurements of human processing times. Lewis and Vasishth (2005) explain in detail several experiments with their ACT-R model and compare them to human data.

Ball's ACT-R Model

Ball et al. (2010), as part of the Synthetic Teammate Project, developed a language processing model implemented in ACT-R that attempts "adherence to well established cognitive constraints." It is based on Ball's "The Double R Theory of Language Comprehension" (Ball 2004). This theory incorporates models of grammar and the comprehension process.

The grammar model represents both "relational and referential meaning" (hence "Double R") in a form that encodes both structure and meaning, as do construction grammars. Although the grammatical notation allows for representing syntax and semantics together, it is loosely based on X-bar theory and does not provide the strong connection between non-lexical items and their meanings that construction grammars typically do.

This model avoids the constraints on short-term memory in ACT-R by adding additional buffers to store intermediate results of parsing. Thus, it avoids using cue-based retrieval as in Lewis's ACT-R model, simplifying the processing, but also straying from modeling human parsing. It uses a bottom-up parsing strategy. Grammar knowledge is stored in declarative memory, and at each step, a production rule creates a cue to retrieve relevant knowledge from that memory. Spreading activation is used to select the item to be retrieved when several items match the cue.

The Rosie Parser

Laird developed a parser in Soar that is embedded within the Rosie robotics system (Kirk, Mininger, and Laird 2016). This semantic parser is based on ideas from NL-Soar and construction grammar. It produces actionable meaning representations that allow the agent to be instructed and directed for a variety of tasks. Lexical and grammatical knowledge are stored in Soar's long-term declarative (semantic) memory. The incremental processing retrieves lexical items from semantic memory, as well as grammatical constructions based on previously retrieved lexical items and derived structures. Grammatical constructions include both semantic mappings from syntactic structures and expectations of future syntactic structures. The word-by-word processing is done with language-independent procedural knowledge. This includes merging expectations with new structures, building up semantic structures based on constructions, and grounding referents in language to the agent's perceptual world model and previously mentioned referents. Soar's chunking mechanism dynamically converts the language processing into rules that move semantic knowledge into procedural memory, leading to better simulated real time performance.

Lucia

Lucia (Lindes and Laird 2016) is a language comprehension system also built within Rosie. In addition to being built on the Soar cognitive architecture, it directly uses ECG to define its linguistic knowledge. An ECG to Soar translator converts a grammar for the Rosie domain, written in the ECG formalism defined by Bryant (2008), into Soar production rules. Lucia then uses these rules to produce the same meanings needed by the Rosie agent as produced by the Rosie parser.

This model goes to the extreme of having all its linguistic knowledge, lexical, grammatical, and semantic, stored in procedural memory. Even though, as in all these systems, this knowledge is hand-written instead of learned, the task of engineering the grammar is easier since it can be written in the ECG language. Additional hand-written production rules to implement grounding, repair, etc. are included, but these are independent of the grammatical knowledge. Almost two thirds of the production rules are generated automatically by the ECG to Soar translator.

Lucia correctly parses all 98 sentences selected from the Rosie corpus attempted to far, as well as a Spanish translation of 50 of these sentences. Parsing proceeds mostly bottom-up, driven by the ECG constructions. Top-down processing is possible for more complex grammars, and is implemented by creating expectations, similar to what is done in the Rosie parser, NL-Soar, and the Lewis' ACT-R parser. In the case of ambiguity, Lucia considers and selects among multiple alternatives, and it does destructive local repair in a way similar to the other incremental models.

The current version of Lucia processes its 98 sentences, using 184 ECG constructions, at a simulated average rate of 132 words/minute. This is within a factor of two of hu-

man processing speed, and likely can be improved when we utilize chunking.

A Case Study: Lucia

Given this convergence of both cognitive modeling and construction grammars in the Lucia model of comprehension, we present details of it below. Additional details are available in Lindes and Laird (2016).

A Simple Example

Consider a simplified description of the processing of the simple sentence in (1).

(1) Pick up the green sphere.

The dynamic comprehension process processes one word at a time using a Soar operator we call comprehend-word. This operator enters a substate in which a lexical-access operator always fires first. This operator puts one or more lexical items on the current state, one for each sense of the current word. For (1) after the word *up* we have an UP item on the top of the stack and a PICK underneath it (upper case names are lexical items). Next a match-construction operator is selected, removing these two items from the stack and replacing them with an instance of the composite PickUp construction.

The next three words, THE, GREEN, and SPHERE, are pushed onto the stack and then two different match-construction operators fire like this:

After *sphere.*	Match SpecPropNoun	Match TransitiveCommand

SPHERE
GREEN
THE
PickUp[
  PICK
  UP
]
	SpecPropNoun[
	  THE
	  GREEN
	  SPHERE
	]
	PickUp[..]
		TransitiveCommand[
		  PickUp[..]
		  SpecPropNoun[..]
		]

The first matches SpecPropNoun, which is a subcase of RefExpr (short for referring expression), and the second combines the verb and its object into a TransitiveCommand, which can be interpreted to tell the agent to perform this action. As each construction is instantiated its corresponding meaning structures are built. These trigger grounding operators that look up the schema for a PickUp action and search the world model to find the object there which matches *the green sphere*. All this information can be used to help resolve ambiguities.

An Example of Attachment and Repair

Now consider a more complicated case where prepositional phrases come into the picture as in (2).

(2) a. Put the green block on the stove.
b. Pick up the green block on the stove.

In (2a) the phrase *on the stove* should modify the verb *put*, giving its target location. In (2b), however, the phrase should modify *the green block* to specify which block to pick up. The key to making this choice is knowledge about whether the verb needs a target location or not.

When Lucia process one of these sentences, a TransitiveCommand construction is built up for everything up through the word *block*. After *on the stove* a PrepPhrase construction is built. At this point an attach-prep-phrase operator examines the context to decide where the phrase should be attached. For (2a) this gives:

After *stove.*	Match ImperativeWithLocation

PrepPhrase[..]
Transitive Command[..]
	ImperativeWithLocation[
	  TransitiveCommand[..]
	  PrepPhrase[..]
	]

For (2b) the verb does not allow an attached target. The attach operator analyzes this case and performs a *snip* operation, which deletes the TransitiveCommand from the stack, and replaces it with its constituents like this:

After *stove.*	After attach-prep-phrase

PrepPhrase[..]
TransitiveCommand[
  PickUp[..]
  SpecPropNoun[..]
]
	PrepPhrase[..]
	SpecPropNoun[..]
	PickUp[..]

This allows a RefExprPrepPhrase construction, another subcase of RefExpr, to be matched, so that after the snip we match two more constructions to complete the sentence:

Match RefExprPrepPhrase	Match TransitiveCommand

RefExprPrepPhrase[
  PrepPhrase[..]
  SpecPropNoun[..]
]
PickUp[..]
	TransitiveCommand[
	  PickUp[..]
	  RefExprPrepPhrase[..]
	]

Once the prepositional phrase is attached to the referring expression, the grounding of that expression is done over again taking into account the new information.

Tanenhaus et al. (1995) performed a human experiment using the sentences in (3).

(3) a. Put the apple on the towel in the box.
b. Put the apple that is on the towel in the box.

The visual scene has two towels, one with an apple and one without, an open box, and distractor. With the ambiguity at *towel* in (3a) human subjects focus on the second

towel, then correct this at *box*. For (3b) this incorrect fixation is not made. When Lucia parses sentences like these, it makes an incorrect attachment of the first prepositional phrase in (3a), then corrects this with a repair. For a sentence like (3b), the incorrect attachment is never made and no repair is needed. This suggests that the local repair process used by Lucia, NL-Soar, and the Rosie parser corresponds to something similar in human processing.

Essential Processes

The Lucia model accomplishes comprehension by a unique algorithm, rooted in the Soar architecture, which interleaves dynamically the several processes needed to produce its results in simulated real time. These processes can be divided into two broad groups: those directly required by the grammatical knowledge, and those needed to integrate with various levels of context.

The grammar processes begin with *recognition*, which finds a match between the current input state and what's on the stack and then constructs an instance of the matched construction. Then *evocation* instantiates the meaning pole of that construction. *Generalization* elaborates these structures with information inherited from constructions and/or schemas in the ECG grammar's subcase of hierarchy. *Unification* of the meaning structures connects different instances to each other according to constraints given in the grammar.

Context effects begin with *selection* among multiple alternatives that suggested by recognition. *Grounding* attaches meaning structures to objects in the agent's world model or ontology of properties, relations, and actions. A prepositional phrase causes *attachment* to choose where to attach that phrase. A *repair* process detects incorrect structures, deletes them, and allows new correct ones to be built.

A key element of the Lucia comprehension theory is that these processes are not sequential phases but rather they are interleaved dynamically as needed in small atomic increments. For example, when a construction is recognized its meaning is immediately evoked and then grounded, so the resulting knowledge is available for subsequent selection or attachment decisions. Thus the order in which operations are performed is not fixed but determined dynamically using the properties of the Soar architecture.

Dealing with Ambiguity

Many situations in natural language are ambiguous in a variety of different ways. Typical parsers try to resolve ambiguities by searching a space of possible parses to find the best fit (eg. Bryant 2008). This approach does not work in an incremental, real-time comprehension system. Lucia uses its *selection*, *attachment*, and *repair* processes to deal with many ambiguous situations. This approach works for many cases and seems promising within the context of a cognitively plausible, embodied comprehension algorithm.

We have seen in (2) how attachment and repair can work for prepositional phrase attachment. A kind of lexical selection can be seen in the sentences in (4).

(4) a. Turn left.
b. Turn on the stove.

In (4a) we simply have the verb *turn* which can be combined with a direction argument to define an action to be performed. Sentence (4b), however, has a completely different sense of *turn*, possibly motivated by turning a nob on a stove to change its state. Here we treat *turn on* as a single grammaticalized lexical item using a lexical construction in ECG whose orthography is set to the whole phrase *turn on*.

In (5) two very different senses of the word *that* are used, as a deictic pronoun and as a relative pronoun.

(5) a. Put that in the pantry.
b. Pick up the green block that is small.

In (5a) we have a simple example of lexical *selection* to choose the deictic meaning of *that* which will be grounded to whatever object is currently being pointed to in the scene. But in (5b) we use all of *selection*, *attachment*, and *repair* for relative *that*, similar to what we saw with (2b).

Sentence (6) shows a more complicated example of lexical ambiguity with two senses of the word *square*.

(6) Put the square in the square box.

This example can be processed using all three processes of *selection*, *attachment*, and *repair.*

Exploring the Architectural Space

Table 1 summarizes of some of the main characteristics of the seven systems we have examined.

Table 1: Summary of model characteristics

	Cognitive architecture	Construction grammar	Incremental processing	Linguistic knowledge
ECG		ECG		
FCG		FCG		
NL-Soar	Soar		Yes	Procedural
Lewis	ACT-R		Yes	Mixed
Ball	ACT-R		Yes	Declarative
Rosie	Soar	Ad hoc	Yes	Declarative
Lucia	Soar	ECG	Yes	Procedural

To build a cognitive model of language comprehension, many different approaches toward using a cognitive architecture are possible. One key dimension relates to linguistic knowledge: how is it represented, where is it stored, and how is it retrieved. Lucia provides an initial example of using ECG as a representation for linguistic knowledge within a cognitive architecture and as a potential model for

human processing. Within the cognitive architecture systems, there is a split between representing linguistic knowledge in either procedural memory (rules) or declarative memory.

If the grammar is represented in declarative rather than procedural memory, how does this affect an incremental parser? Initially a declarative representation is more flexible because there is not a single method for accessing the knowledge (matching rule conditions), but it is less efficient because every construction has to be deliberately retrieved from declarative memory and then instantiated before it can be used. However, if comprehension operators are chunked as in NL-Soar, the agent will, with experience, convert this declarative knowledge into efficient procedural knowledge. Nevertheless, having constructions in declarative memory may make it much easier for future work on acquiring linguistic knowledge from experience. Lewis's ACT-R model stores all lexical knowledge in declarative memory, but still uses procedural memory for all grammatical knowledge.

In Lucia, we directly compile the grammar from a declarative ECG representation into production rules, similar to the full procedural knowledge built by chunking in NL-Soar. This has helped experiment with whether ECG is a good representation for grammar in a cognitive model. We are furthering this research by also experimenting with translating ECG into a declarative representation in Soar's semantic memory instead of into production rules. Our hypothesis is that spreading activation will allow construction and lexical retrieval to be dependent on the context as defined by the contents of working memory. Ball's ACT-R parser uses a single-depth spread, but in Soar, we are able to use much deeper spreads, potentially using more indirect associations to influence retrieval in ambiguous cases. Initially these retrievals will slow the comprehension process, but chunking has the potential to convert declarative knowledge into procedural knowledge, and real-time comprehension.

Another architectural issue is connecting new information into the developing intermediate result of parsing. NL-Soar, the Rosie parser, and Lucia all use explicit data structures in working memory and production rules to match items that need to be connected. Lewis's ACT-R model uses a different method called cue-based retrieval. When each word is processed, expectations are created and stored in declarative memory, which acts as both a short-term and long-term memory. Later words (and derived structures) are used as cues to retrieve a previous expectation that is then combined with the new structure. Lewis and Vasishth (2005) show that this approach produces behavior that matches human timing data fairly well. In Soar, however, short-term results are held in working memory, and working memory has no cue-based retrieval mechanism. Given the success of cue-based retrieval, it is worth considering whether some modifications to Soar are needed to accommodate it.

Both Bryant (2008) and Jurafsky (1996) annotate constructions in their grammars with probabilities as a way of resolving ambiguities. These probabilities are derived from off-line statistical processing of a corpus. Although statistics certainly play a role in human language processing, the cognitive models we have discussed do not use probabilistic grammars. Instead they use local context, both situational and linguistic, to resolve ambiguities. When declarative memories are used, this is done by biasing retrievals from long-term memory. Embedded within that retrieval process is the use of activation that incorporates recency and frequency (base-level activation), and historic co-occurrence of both lexical and grammatical constructions (spreading activation). We are exploring how to structure the parsing processes and long-term memory to best take advantage of both base-level and spreading activation.

None of these models deals with the problem of acquiring new linguistic knowledge. However, there is much psychological research in this area (e.g. Tomasello 2003; Krashen 2003). Research on acquisition has also been done in both the ECG (Chang 2008; Mok 2008) and FCG (Garcia-Casademont and Steels 2015, 2016) communities. Starting with constructions represented in declarative memory will probably be important in extending this research into cognitive models of comprehension. New constructions can be added and existing ones modified in this memory, with chunking making it efficient in real time.

Conclusions

In this paper we have briefly reviewed research showing that models based on the general capabilities of a modern cognitive architecture can be used to comprehend natural language, that this comprehension can take place in simulated real time, and that it can be grounded in the perception and action capabilities of an embodied agent. We have also explored using Embodied Construction Grammar (ECG) for representing the linguistic knowledge in a cognitive model of comprehension, demonstrating that useful comprehension can be achieved, including resolving some kinds of ambiguity. This system uses ECG in an incremental processing model that does not use holistic optimization over a whole utterance or statistical knowledge gathered off line from a large corpus.

Considering the current state of the art, many questions for future research emerge. What is the best way to apply the different capabilities of a cognitive architecture to the language comprehension problem? What does human data tell us about an architecture and how to apply it? Are there enhancements to an architecture that will allow it to better model human comprehension? Are there extensions to construction grammar formalisms and their implementa-

tion in cognitive models that will help expand the coverage of these models to a wider range of natural language with all its ambiguities and ungrammaticalities? How well can we achieve effective communication between humans and embodied agents using these models? Finally, how can these models acquire their linguistic knowledge from their experience? Many challenges lie ahead.

Acknowledgments

The work described here was supported by the Air Force Office of Scientific Research under Grant Number FA9550-15-1-0157. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the AFOSR or the U.S. Government.

References

Anderson, J. R.; Byrne, M. D.; Douglass, S.; Lebiere, C.; and Qin, Y. 2004. An Integrated Theory of the Mind. *Psychological Review 111*, 1036-1060.

Ball, J. T. 2004. *The Double R Theory of Language Comprehension*. United States Air Force Research Laboratory report AFRL-HE-AZ-TR-2003-0110.

Ball, J.; Freiman, M; Rodgers, S.; and Myers, C. 2010. Toward a Functional Model of Human Language Processing. Presented as a poster at *32nd Annual Conference of the Cognitive Science Society*. Portland, OR.

Bergen, B. and Chang N. 2013. Embodied Construction Grammar. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*, 168-190. New York: Oxford University Press.

Bryant, J. E. 2008. *Best-Fit Constructional Analysis*. Ph.D. diss., Computer Science, University of California, Berkeley, CA.

Chang, N. C. 2008. *Constructing grammar: A computational model of the emergence of early constructions*. Ph.D. diss., Computer Science, University of California, Berkeley, CA.

Eppe, M.; Trott, S.; Raghuram, V.; Feldman, J.; and Janin, A. 2016. Application-Independent and Integration-Friendly Natural Language Understanding. *GCAI 2016. 2nd Global Conference on Artificial Intelligence.* Easy Chair Publications.

Feldman, J.; Dodge, E.; and Bryant, J. 2009. Embodied Construction Grammar. In *The Oxford Handbook of Linguistic Analysis*, Bernd Heine and Heiko Narrog eds. Oxford Handbooks Online.

Garcia-Casademont, E. and Steels, L. 2015. Usage-based Grammar Learning as Insight Problem Solving. *CEUR Workshop Proceedings* 1419: 258-263.

Garcia-Casademont, E. and Steels, L. 2016. Insight Grammar Learning. *Journal of Cognitive Science* 17-1: 27-62.

Goldberg, A. E. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. The University of Chicago Press.

Goldberg, A. E. 2013. Constructionist Approaches. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*, 15-31. New York: Oxford University Press.

Hoffman, T., and Trousdale, G., eds. 2013. *The Oxford Handbook of Construction Grammar*. New York: Oxford University Press.

Jurafsky, D. 1996. A Probabilistic Model of Lexical and Syntactic Access and Disambiguation. *Cognitive Science 20*, 137-194.

Just, M. A. and Carpenter, P. A. 1987. *The Psychology of Reading and Language Comprehension*. Boston, MA: Allyn and Bacon.

Kirk, J.; Mininger, A.; and Laird, J. 2016. Learning task goals interactively with visual demonstrations. *2016 Annual International Conference on Biologically Inspired Cognitive Architectures*, New York City, NY.

Krashen, S. D. 2003. *Explorations in Language Acquisition and Use: The Taipei Lectures*. Portsmouth, NH: Heineman.

Laird, J. E. 2012. *The Soar Cognitive Architecture*. Cambridge, MA: The MIT Press.

Lehman, J. F.; Lewis, R. L.; and Newell, A. 1991. Integrating Knowledge Sources in Language Comprehension. *The Soar Papers, Vol. II*. Cambridge, MA: MIT Press.

Lewis, R. L. 1993. *An Architecturally-based Theory of Human Sentence Comprehension*. Ph.D. diss., Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Lewis, R. L. and Vasishth, S. 2005. An Activation-Based Model of Sentence Processing as Skilled Memory Retrieval. *Cognitive Science 29*, 375-419.

Lewis, R. L.; Vasishth, S.; and Van Dyke, J. A. 2006. Computational principles of working memory in sentence comprehension. *TRENDS in Cogntive Science* 10(10): 447-454.

Lindes, P. and Laird. J. E. 2016. Toward Integrating Cognitive Linguistics and Cognitive Language Processing. *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016)*. Penn State: University Park, PA.

Lonsdale, D.; McGhee, J.; Glenn, N.; and Anderson, T. (2011). *The XNL-Soar Sandbox*. Presentation at The 31st Soar Workshop, June 13-17, 2001, University of Michigan.

Mok, E. H. 2008. *Contextual Bootstrapping for Grammar Learning*. Ph.D. diss., Computer Science, University of California, Berkeley, CA.

Newell, A. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Rubinoff, R. and Lehman, J. F. 1994. Real-time Natural Language Generation in NL-Soar. *7th International Generation Workshop*, Kennebunkport, Maine.

Steels, L. 2013. Fluid Construction Grammar. In Thomas Hoffman and Graeme Trousdale, eds, *The Oxford Handbook of Construction Grammar*. Oxford University Press, New York, pp. 153-167.

Steels, L.; De Beule, J.; and Wellens, P. 2012. Fluid Construction Grammar in Real Robots. In Steels and Hild eds. *Language Grounding in Robots*. Springer.

Steels, L., and Hild, M. eds. 2012. *Language Grounding in Robots*. Springer.

Steels, L. and van Trijp, R. (2011). How to Make Construction Grammars Fluid and Robust. In Luc Steels ed. *Design Patterns in Fluid Construction Grammar*, 301-330. Amsterdam: John Benjamins.

Tanenhaus, M. K.; Spivey-Knowlton, M. J.; Eberhard, K. M.; and Sedivy, J. C. 1995. Integration of Visual and Linguistic Information in Spoken Language Comprehension. *Science* 268:1632-1634.

Tomasello, M. 2003. *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Cambridge, MA: Harvard University Press.

Ullman, M. T. 2004. Contributions of memory circuits to language: the declarative/procedural model. *Cognition 92*, 231-270.

Van Dyke, J. A. and Lewis, R. L. 2003. Distinguishing effects of structure and decay on attachment and repair: A cue-based parsing account of recovery from misanalyzed ambiguities. *Journal of Memory and Language 49*, 285-316.