Formalizing Langacker’s Notions of Nouns and Verbs

Introduction

Langacker’s 1990 book, *Concept, Image, Symbol: The Cognitive Basis of Grammar*, is rich in linguistic insight, providing the basic conceptual underpinnings of a cognitively realistic grammar. Unfortunately, most of the work in the book is not precise enough to be of immediate use for language understanding systems. This lack of precision is the result of two shortcomings: his use of a pictorial representation language, and second, the lack of a well-defined framework within which rigorous definitions can be provided.

The Neural Theory of Language (NTL) group seeks to overcome these limitations by formalizing Langacker’s work within the theoretical framework of simulation-based language understanding and construction grammar. The simulation-based model of language understanding splits language understanding into an analysis phase and an enactment phase. The analysis phase uses construction grammar to map utterances onto static representations of meaning while the enactment phase deals with the dynamics. Within this framework, NTL provides precise representation languages for describing grammar, semantics and both dynamic and static conceptual structures.

Using the conceptual substrate provided by NTL, it is possible to provide rigorous definitions for Langacker’s ideas. This paper concentrates on formalizing Langacker’s ideas about nouns and verbs in chapter three of his book.

The Neural Theory of Language

Within the NTL framework, language understanding is separated into two conceptually distinct phases: analysis and enactment. The analysis phase takes the utterance as input and uses constructions, conceptual and semantic structure, and context to produce a static representation of that utterance’s meaning. This static representation is called a semantic specification or *semspec*.

The *semspec* contains a set of bindings between the participants and their roles in the scene. The *semspec* is said to *parameterize* the enactment phase because given these bindings and the associated x-schema, the utterance can be simulated. The simulation takes the utterance’s temporal structure into account and produces both dynamic and context-dependent inference about the communicated event.

The NTL framework also provides representation languages to describe the linguistic and conceptual knowledge that goes into analysis and enactment. Not only do these representation languages possess the rigor necessary to build computational implementations, but they also have clear conceptual reductions to cognitive structure.

The representation language used for the analysis phase is called Embodied Construction Grammar [1]. Embodied Construction Grammar (ECG) is a cognitively-motivated, construction-based formalism that leverages the precision of unification grammar [8] to deliver a formalism that can be implemented on a computer. It has representational mechanisms for describing constructions and configurational knowledge like frames [5] and image schemas [6]. Researchers have already taken advantage of ECG’s computational precision by building implementations of constructional analysis [2] and child language acquisition [4].

The representation language used for the enactment phase is an extension to stochastic petri nets called Executing Schemas [7]. Intended to precisely describe human motor control, executing schemas (x-schemas) are dynamic models of actions that can model parallel processing, iteration and different levels of abstraction within a particular action. X-schemas have successfully been used to model linguistic aspect [3, 7] as well as metaphor understanding [7].

The Into Construction

This section provides a brief introduction to the ECG formalism\(^1\) as well as giving a basic example of how to incorporate Langacker’s ideas into a precise framework.

---

\(^1\)It assumes familiarity with unification-based grammar.
Langacker describes a preposition as an atemporal relation between the participants in an utterance. Within these atemporal relations, there is an asymmetrical relationship between these participants in that the focused entity’s position (the trajector) is described with respect to another (the landmark).

Figure 1 shows the conceptual structure (the Container, SPG and Trajector-Landmark schemas\(^2\)) involved in defining \emph{into} as well as the lexical Into construction. The construction makes the mapping of form to meaning explicit with blocks for the form pole (\emph{form}) and the meaning pole (\emph{meaning}). The form and meaning blocks constrain their domains in accordance with the construction. For example, To state that the notion of a container is involved, the landmark role is constrained to be of type Container.

The Into construction also \emph{evokes} an SPG schema, giving it the local name \(s\), to mark the dynamic aspect of \emph{into}. The relationship between this dynamic aspect of \emph{into} and the notion of the container is captured by the binding constraints that link the \(s\.source\) to the \(landmark\.exterior\) and the \(s\.goal\) to the \(landmark\.interior\). The evokes operator is conceptually similar to neural activation of related concepts.

\textbf{Profiled vs. Designated Meaning in a Construction}

Langacker distinguishes between a construction’s profiled meaning and its designated meaning. In the ECG formalism, a structure is profiled if it is present in the meaning pole. In the case of Into, it profiles the SPG, Trajector-Landmark, and Container schemas. It only designates the Trajector-Landmark relation, however, because this is the structure bound to the construction’s meaning pole.

\textbf{Simple vs. Complex Prepositions}

The preposition \emph{into} is a complex preposition in Langacker’s terminology because (much like dynamic \emph{across},) it describes a chain of states varying through time that are viewed atemporally as a gestalt. The use of the SPG schema provides a concrete way to refer to this gestalt without having to make any distinctions between \emph{simple} or \emph{complex} prepositions. The SPG schema is thus a shorthand for a chain of states, and these states become accessible when the SPG is paired up with a motion verb during simulation.

\textbf{Nouns}

Langacker uses the beginning of chapter three to provide the basic semantic hierarchy for the designatum of nouns. The basic intuition that he develops is that nouns designate regions. More specifically, count nouns designate bounded regions with respect to some larger domain and mass nouns designate unbounded regions.

\(^2\)Meaning is represented in ECG by schemas. These schemas are similar to the feature structures in unification grammars in that they have roles and identification constraints.
schema Region
  schematic
  subcase of Region
  evokes Region as domain

schema Intrinsically-Bounded-Region
  schematic
  subcase of Region
  evokes Region as domain

schema Spatially-Bounded-Region
  schematic
  subcase of Intrinsically-Bounded-Region
  evokes Spatially-Bounded-Region as domain

Figure 2: Some of the schemas that act as designata for mass and count nouns.

countconstruction MassNoun
countsubcase of Noun
  meaning : Unintrinsically-Bounded-Region
countconstruction CountNoun
countsubcase of Noun
  meaning : Intrinsically-Bounded-Region

Figure 3: The abstract construction acting as supertypes for mass and count nouns.

To represent such distinctions, it is necessary to provide an abbreviated semantic hierarchy consistent with Langacker’s ideas. Such a hierarchy is easily expressed with ECG. Figure 2 shows some of the schemas associated with count and mass nouns.

The schemas in figure 2 are arranged into an inheritance hierarchy using the subcase of keyword. In the case of this hierarchy, the schematic Region type is the supertype of Intrinsically-Bounded-Region and Unintrinsically-Bounded-Region. Unintrinsically-Bounded-Region is the supertype corresponding to mass nouns because while substances like butter and water are not bounded to start with, they can be bound within tubs and bottles, respectively. The type Intrinsically-Bounded-Region, however, is associated with count nouns. As such, this schema evokes a schematic domain (itself a Region) within which it is bound. One of Intrinsically-Bounded-Region’s subtypes is Spatially-Bounded-Region which further constrains the evoked domain to itself be of type Spatially-Bounded-Region since the bounding domain of a spatial entity must also be a spatial entity. Figure 3 shows how Intrinsically-Bounded-Region and Unintrinsically-Bounded-Region are the designata of the CountNoun and MassNoun construction, respectively.

Figure 4 shows a concrete example of a count noun for the concept of nose. As Langacker points out, such concepts are ill-defined without their domain. In this case, the domain is the Face schema (not shown). To make the domain accessible, the Nose schema does two things. First it evokes the Face schema as its domain. Then it binds itself to the domain using the self operator. The self operator lets a construction or schema refer to itself, thereby making it possible for that schema or construction to bind itself to a larger frame. So with the self operator, the Nose schema binds itself to a role within that frame, fitting itself within the bounding concept of Face. Because the NoseCxn uses the Nose schema as its designatum, the Face schema is automatically profiled in the NoseCxn.

Figure 4: The construction and schema for the count noun nose. Since the complete schema and construction hierarchies are not defined, the direct supertypes of each of these structures are not shown, but the Nose schema inherits (indirectly) from Intrinsically-Bounded-Region and NoseCxn (indirectly) inherits from the CountNoun construction.
Figure 5: A generic x-schema that allows iteration and interrupted termination. X-schemas can model actions at varying levels of abstraction. Thus subsidiary actions can be expanded into their own substructures, as indicated by the dashed hexagon around a set of actions and states.

**Verbs**

Langacker describes (finite) verbs as profiling processes while prepositions profile atemporal relations. We have already seen how a preposition can specify an atemporal relation using ECG, but providing a precise notion of process requires introduction to the x-schema formalism.

An x-schema is a parameterized model of an action. Figure 5 shows the states associated with a generic x-schema. The circles denote states internal to this action while the hexagons denote subsidiary actions internal to this action. Particular actions will instantiate these generic states and actions differently. For example, the Iterate action for the Walk x-schema is taking a step.

The generic action starts out in the Enabled state, and if the Prepare action succeeds, the action enters the Ready state. From the Ready state the actual body of the action can be Started after which the action is in its Ongoing state. In the Ongoing state, there are three possible continuations, one for repeating Iterate, one for cutting the action short (Interrupt), and one for reaching the goal (Finish).

Different actions use different parts of this generic x-schema structure depending on their aspectual structure. With the x-schema (and parameterization) in hand, the simulator can model the flow of activation through the x-schema. Activation is represented as resource tokens moving around the arcs of the x-schema. Tracking this flow of resource tokens is the simulation process.

With a precise notion of a process, we can finally provide a notion of the semantics of finite verbs in a way consistent with Langacker: Verbs designate x-schemas. Now we can easily define Langacker’s notions of summary and sequential scanning. Sequential scanning is equivalent to simulating the action, while summary scanning is the collection of states that the x-schema went through during simulation.

Using x-schemas, the different aspectual classes that Langacker describes are distinguished structurally. For example, with Langacker’s notion of imperfective, a verb like resemble is an x-schema that iterates indefinitely because no resources are lost at each step of iteration. Contrast this with a verb like run, in which resources are used up on each step of the iteration. Langacker’s notion of perfective verbs can easily be defined as x-schemas that have distinct, self-contained substructures.

---

3Copula *be* designates an x-schema with just a single state. This is how utterances like *John is tall* can profile a process but still have the character of a predicate.

4In computer science terminology, this would be described as an execution trace.
reachable Finish states.

The semantics of verbal morphology can also be handled within the simulation framework. For example, a stative participle like broken designates the Done state of the Break x-schema. Another example is the progressive ending -ing which designates the transition between the Ready state and the Done state.

Abstract Nouns

The final topic covered in chapter three is Langacker’s approach to abstract nouns. He treats abstract nouns as the linguistic handle referring to the result of a summary scan on a process. In our framework, an abstract noun like explosion would evoke the Explode x-schema, and then bind itself to the collection of states that the x-schema went through during the simulation of an Explode action.

References


