The logic of words: a monadic decomposition of lexical meaning

Chenchen Song, Zhejiang University, cjs021@zju.edu.cn

Words are the building blocks of human language. A most noticeable aspect of word meanings is their idiosyncrasy. Thus, it is an arbitrary fact that the word *dog* means 'a common four-legged animal' instead of 'a musical instrument with a row of strings'. Given the idiosyncratic nature of words, studies of their meanings have long been kept apart from studies of sentence meanings. In the latter area, idiosyncratic word meanings are often just treated as atomic constants, as in the following run-of-the-mill logical forms (taken from Heim & Kratzer 1998:44):

(1) $[\![\text{smokes}]\!] = \lambda x \in D_e$. x smokes, $[\![\text{loves}]\!] = \lambda x \in D_e$. [$\lambda y \in D_e$. y loves x], $[\![\text{Ann}]\!]$ = Ann
(e notates the individual type, and D_e notates the domain of individuals)

When the words *smokes*, *loves*, and *Ann* are translated into logical forms, their idiosyncratic aspects remain intact, and what the λ-terms render explicit is merely their semantic types. This is normal practice in model-theoretic semantics, which is preserved in recent developments that are otherwise driven by lexical decomposition, as in the neo-Davidsonian representation in (2).

(2) $[\![\text{Jones buttered the toast}]\!] = \exists e[\text{BUTTER}(e) \wedge \text{AG}(e)=j \wedge \text{TH}(e)=t]$ (Landman 2000:1–2)
(e notates the event variable, and AG/TH notate the thematic roles Agent/Theme)

While a verb's arguments are severed out of its logical form in neo-Davidsonian semantics, its idiosyncratic aspect (e.g., that *butter* means 'a buttering event' rather than 'a singing event') is still treated as an integral part of the typed variable–introducing λ-term, though the idiosyncrasy plays no role in subsequent type logical computation and is just carried along till the sentence level. Against this backdrop, in this paper I present a method to further sever the idiosyncratic part out of a word's semantic representation, thus pushing the lexical decomposition spirit to the maximum. The method uses the category-theoretic notion monad, and the semantic theory thus developed matches an independently developed syntactic theory called root syntax.

Root syntax Root syntax (Halle & Marantz 1993, Borer 2005, et seq.) is a currently popular branch of Chomskyan generative grammar (see Chomsky 1995 for the background theory). Its core idea is to separate the idiosyncratic and the categorial aspect of word formation in syntactic representation. The former is encoded in a "root" while the latter is a category-defining object. I illustrate root syntax with the following phrase structure schema (which is equivalent to a tree):

(3) $[_X\ X\ \sqrt{}\ ]$ (X is a syntactic category, $\sqrt{}$ is a root, and their merger corresponds to a word)

For example, the noun *dog* is assigned the syntactic representation $[_N\ N\ \sqrt{\text{DOG}}]$, and the verb *love* is assigned $[_V\ V\ \sqrt{\text{LOVE}}]$. When the categorial context changes, the same root may get a different interpretation (and so become a different word), as in the verb *dog* $[_V\ V\ \sqrt{\text{DOG}}]$ and the noun *love* $[_N\ N\ \sqrt{\text{LOVE}}]$. A root, being categoryless, encodes the highly vague information that underlies all its potential categorized meanings (e.g., the information in $\sqrt{\text{LOVE}}$ is the conceptual "nebula" that could be shaped into both the nominal concept 'love' and the verbal concept 'to love').

Root semantics Despite the popularity of root-oriented thinking in formal syntax in the past two decades, the idea has had surprisingly little impact on formal semantics. This leaves a hole in the syntax-to-semantics mapping. The monadic method presented here can help fill that hole. With a usual λ-calculus-based metalanguage for natural language semantics, we can define a cartesian closed category **Syn** of semantic types, which itself is a tiny subcategory of the category of sets. Next, we define a monad ⟨T, η, μ⟩ on **Syn** as follows:

(4) For any **Syn**-object A, TA ≜ ⟨A, {⟨X, $\sqrt{_1}$⟩, ⟨Y, $\sqrt{_2}$⟩, …}⟩, which is a pair of A and a set of category-root pairs, with X, Y, … occurring in A. For any **Syn**-arrow f, Tf ≜ λ⟨x, Q⟩. ⟨f(x), Q⟩, where Q stands for a set of category-root pairs. Let H and R be the set of syntactic categories and

the set of roots in a natural language, then a set of category-root pairs is a member of the power set $\mathbf{P}(H\times R)$, and the endofunctor $T \triangleq - \times \mathbf{P}(H\times R)$ sends each semantic type to a product type. The natural transformations η and μ are defined as $\eta_A x \triangleq \langle x, \emptyset\rangle$, which embeds an element of any type A in a trivial monadic context TA, and $\mu_A(\langle\langle x, P\rangle, Q\rangle) \triangleq \langle x, P \cup Q\rangle$, which joins two sets of category-root pairs into a single set and thereby reduces a double-layered monadic context TTA to a single-layered one TA. With μ, we further define a bind operation >>= on an element of type TA and a function f of type $A \rightarrow TB$ as $TA>>=f \triangleq \mu_B(Tf(TA))$, which yields an element of type TB. Finally, we define an ancillary operation **write**$(\langle X, \surd\rangle) \triangleq \langle 1, \{\langle X, \surd\}\rangle$, which takes a single category-root pair and embeds it in a trivial monadic context, where 1 is a placeholder.

The above-defined monad is basically the writer monad in functional programming, in which respect the current work heavily builds on Asudeh & Giorgolo's (2020) use of the writer monad to model conventional implicature in certain lexical words (e.g., the negative attitude in *Yank*). The parallelism between conventional implicature and root idiosyncrasy is that both are non-truth-conditional but conventionalized information (in fact the former $\subset$ the latter), and in both cases the monad keeps compositional and noncompositional meanings separate. Specifically, in our case it lets "pure" computation normally proceed while keeping a record of the roots in the syntax-to-semantics mapping, logging which syntactic categories are tagged by what roots. This record is crucial for the final, overall interpretation of natural language sentences, where formal-logical and idiosyncratic information are both referenced. See (5) for an illustration.

(5) $[\![[_N\ N\ \surd_{DOG}]]\!] = \mathbf{write}(N_{\surd DOG}) >>= \lambda y.\eta[\![N]\!] = \langle[\![N]\!], \{\langle N, \surd_{DOG}\rangle\}\rangle$
(an entity that is idiosyncratically characterized by the root $\surd$DOG)

Generalized root syntax A key merit of the monadic semantics presented here is that it can naturally accommodate a further development of root syntax—called generalized root syntax in Song (2019)—which uses the root categorization schema in (3) not only for lexical words (e.g., nouns) but also for semilexical words (e.g., classifiers), which have both lexical and grammatical contents. Such words are widely observed in natural languages, and their root-syntactic analysis is simple: we just fill the X in (3) with a grammatical category instead of a lexical one.

(6) Ex.: Vietnamese has multiple negation particles, whose usage is conditioned by register, speaker attitudes, etc. Below is a root-based analysis of the highly informal negator *cóc* :
$[\![[_{Neg}\ Neg\ \surd_{CÓC}]]\!] = \mathbf{write}(Neg_{\surd CÓC}) >>= \lambda y.\eta[\![Neg]\!] = \langle[\![Neg]\!], \{\langle Neg, \surd_{CÓC}\rangle\}\rangle$
(a negation operator that is idiosyncratically characterized by the root $\surd$CÓC)

Categorical linguistics Finally, I want to make a connection between the current work and the broader area of categorical linguistics. The "root spirit" here, namely the separate treatment and systematic unification of compositional and idiosyncratic information in theoretical linguistics, is reminiscent of the treatment of compositional and lexical meanings in Coecke et al.'s (2010 et seq.) more NLP-oriented compositional distributional semantics. Thus, their type-tagged vector spaces are similar to our root-tagged types (despite the disparate grammatical theories), as in (7).

(7) a. $(V\otimes S\otimes W, n^r s n^l)$ (the vector space–based meaning space for transitive verbs)
b. $\langle[\![V]\!], \{\langle V, \surd\rangle\}\rangle$ (the monadic semantics of verbs)

Leaving aside the grammatical theory–induced differences, we see that Coecke et al.'s meaning space largely corresponds to our log space, and their type tag, to our logical form. Since both lines of work use category theory, this cross-framework connection is worth further exploration.

Selected references **Asudeh A. & G. Giorgolo (2020).** *Enriched meanings.* OUP. **Coecke, B., M. Sadrzadeh & S. Clark (2010).** Mathematical foundations for a compositional distributional model of meaning (arXiv:1003.4394). **Heim, I. & A. Kratzer** (1998). *Semantics in generative grammar.* Blackwell. **Song, C. (2019).** On the formal flexibility of syntactic categories, University of Cambridge dissertation.