Sense abstraction: a unified framework of intensionality, predicate abstraction, and alternative semantics

1 Introduction

In his landmark paper *On sense and reference*, Frege discusses two different layers to the meanings of natural language expressions, which he calls "sense" and "reference". Reference, or extensional meaning, is the real-world object or value to which a linguistic expression points. This can be an entity (the referent of "Snorri Sturluson" is the medieval Icelandic author Snorri Sturluson), although in many cases it is less tangible (the referent of "four is an odd number" is the symbol false). On the other hand, sense, or intensional meaning, is concerned with the logical structure of an expression and the algorithm which is needed to determine its referent. One case where the concept of sense is particularly apparent is in expressions with undefined referents. For example, "the largest natural number" has no referent, but one can still grasp its sense intuitively.

In modern compositional semantics, the difference between the truth conditions and the truth value of a sentence is a special case of Frege's sense-reference distinction [1]. A logical formula representing a sentence's truth conditions can be thought of as the sentence's sense which, when evaluated in some world, gives us the referent: true, false, or, possibly, undefined. This paper is motivated by the observation that one can generalize a sense semantics for all natural language expressions using t-typed sentences as an example. By the principle of compositionality, a sentence's LF-tree is a computational graph for its truth conditions; *i.e.* when we evaluate the truth conditions of a sentence, we can carry out this computation by recursively composing meanings in the LF-tree from the leaves up to the root. Therefore, since a sentence's truth conditions are determined by its LF-tree, we define the sense of a sentence as the LF-tree rather than the truth conditions themselves.

This is useful because all linguistic constituents within a sentence have an LF-subtree, so we can generalize that the sense of any linguistic expression is its own LF-subtree. Since we will represent subtrees by their root nodes, the sense of a node is the unevaluated node itself. On the other hand, we will define the referent of a node as its normal denotation obtained from recursively evaluating its subtree.

Definition 1.1 (Sense)**.** *For a node* α*,* $\text{sense}(\alpha) = \alpha$.

Definition 1.2 (Reference)**.** *For a node* α*,* $\text{ref}(\alpha, w) = [\![\alpha]\!]^w$.

1.1 Sense abstraction

As Frege points out, the referents of some natural language constructions depend on the sense of their sub-constituents [2]. For example, this is often true of English "that" clauses: "I said that the year is 1738" could be true in 2016 even though the referent of "the year is 1738" is false in the present.

In order to capture this with our sense formalism, we will introduce a new semantic type l for LF-subtrees. We will also define $[\![that]\!]$ as some symbol SENSE, which will interact with the following compositional rule:

Definition 1.3 (Sense abstraction)**.** *If α has children $\{\beta,\gamma\}$ such that $[\![\beta]\!] = \text{SENSE}$, then $[\![\alpha]\!]_l = \text{sense}(\gamma) = \gamma$.*

Such a system lets us straightforwardly compute the truth conditions of sentences like "I believe dragons exist". As will be shown, it also gives us a unified treatment of intensionality, predicate abstraction, and alternative semantics.

1.2 LF-trees formalized

It will be useful to have a formal definition of LF-subtrees. We will consider LF-subtrees to be recursive data structures made of tuples.

Definition 1.4 (LF-leaf)**.** *A leaf node is a tuple (σ, f, Λ) where σ is the node's label, $f = 1 \iff$ the node is f-marked, and Λ is the lambda expression giving the meaning of the node in the lexicon.*

Definition 1.5 (LF-parent)**.** *A parent node is a tuple $(\sigma, f, \beta, \gamma)$ where σ is the node's label, $f = 1 \iff$ the node is f-marked, and β and γ are the left and right child nodes respectively. If γ is null, then β is the sole child.*

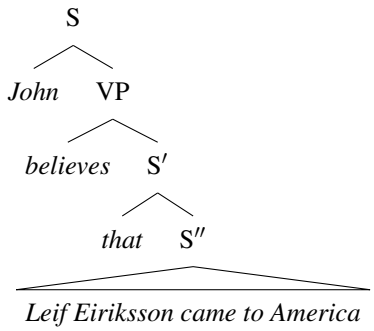
Definition 1.6 (LF-subtree)**.** *An LF-subtree is any syntactic node that is either an LF-leaf or an LF-parent.*

2 Intensionality

To show how sense abstraction can be used to handle intensionality, we will consider the example sentence "John believes that Leif Eiriksson came to America". Let $\mathbb{B}$ be the function which takes an individual x and a world w and returns the set of worlds that x living in w believes are possible [3]. We can define the denotation of "believe" in terms of $\mathbb{B}$.

Definition 2.1 (Denotation of "believe")**.**

$$[\![believe]\!]^w_{\langle l,\langle e,t\rangle\rangle} = \lambda\alpha_l.\lambda x.\forall w' \in \mathbb{B}(x)(w) : ([\![\alpha]\!]^{w'} = 1)$$



$$[\![\mathrm{S}']\!]^w_l = \mathrm{S}''$$

$$[\![\mathrm{VP}]\!]^w_{\langle e,t\rangle} = [\![believes]\!]^w([\![\mathrm{S}']\!]^w) = \big(\lambda\alpha_l.\lambda x.\forall w' \in \mathbb{B}(x)(w) : ([\![\alpha]\!]^{w'} = 1)\big)(\mathrm{S}'')$$

$$= \lambda x.\forall w' \in \mathbb{B}(x)(w) : ([\![\mathrm{S}'']\!]^{w'} = 1)$$

$$= \lambda x.\forall w' \in \mathbb{B}(x)(w) : \text{Leif Eiriksson came to America in } w'$$

$$[\![\mathrm{S}]\!]^w_t = [\![\mathrm{VP}]\!]^w([\![Will]\!]^w)$$

$$= \big(\lambda x.\forall w' \in \mathbb{B}(x)(w) : \text{Leif Eiriksson came to America in } w\ \big)(\text{John})$$

$$= 1 \iff \forall w' \in \mathbb{B}(\text{John})(w) : \text{Leif Eiriksson came to America in } w'. \quad \square$$

3 Predicate abstraction

Sense abstraction eliminates the need for a special compositional rule for predicate abstraction. Instead, we can arrive at the same LF structures using a lexicalized version of the old rule. We will introduce a type z for numerical indexes and provide the following denotation for "who".

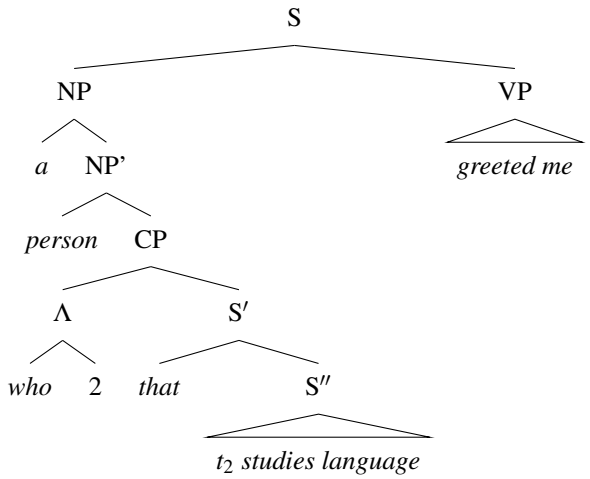
Definition 3.1 (Denotation of "who")**.** [1]

$$[\![who]\!]^g_{\langle z,\langle l,\langle e,t\rangle\rangle\rangle} = \lambda i_z.\lambda\alpha_l.\lambda x.[\![\alpha]\!]^{[i\to x]||g}$$

3.1 Relative clauses

With this sense-sensitive denotation for "who", we can derive the truth conditions for "A person who studies language greeted me" without a special rule like predicate abstraction.

[1] Where $[i \to x]||g$ is the function that maps i to x and returns $g(n)$ for any other n.



$$[\![\mathrm{S}']\!]^{w,g}_{l} = \mathrm{S}''$$

$$[\![\Lambda]\!]^{w,g}_{\langle l,\langle e,t\rangle\rangle} = [\![who]\!]^{w,g}([\![2]\!]^{w,g}) = \big(\lambda i_z.\lambda\alpha_l.\lambda x.[\![\alpha]\!]^{w,[i\to x]||g}\big)(2)$$
$$= \lambda\alpha_l.\lambda x.[\![\alpha]\!]^{w,[2\to x]||g}$$

$$[\![\mathrm{CP}]\!]^{w,g}_{\langle e,t\rangle} = [\![\Lambda]\!]^{w,g}([\![\mathrm{S}']\!]^{w,g}) = \big(\lambda\alpha_l.\lambda x.[\![\alpha]\!]^{w,[2\to x]||g}\big)(\mathrm{S}'') = \lambda x.[\![\mathrm{S}'']\!]^{w,[2\to x]||g}$$
$$= \lambda x.([2\to x]||g)(2) \text{ studies language} = \lambda x.x \text{ studies language}$$

$$[\![\mathrm{NP}']\!]^{w,g}_{\langle e,t\rangle} = \lambda y.[\![person]\!]^{w,g} \wedge [\![\mathrm{CP}]\!]^{w,g}$$
$$= \lambda y.(\lambda x.x \text{ is a person})(y) \wedge (\lambda x.x \text{ studies language})(y)$$
$$= \lambda y.(y \text{ is a person}) \wedge (y \text{ studies language})$$

$$[\![\mathrm{NP}]\!]^{w,g}_{\langle\langle e,t\rangle,t\rangle} = [\![a]\!]^{w,g}([\![\mathrm{NP}']\!]^{w,g})$$
$$= \big(\lambda p_{\langle e,t\rangle}.\lambda q_{\langle e,t\rangle}.\exists x(p(x)\wedge q(x))\big)\big(\lambda y.(y \text{ is a person}) \wedge (y \text{ studies language})\big)$$
$$= \lambda q_{\langle e,t\rangle}.\exists x\big((x \text{ is a person}) \wedge (x \text{ studies language}) \wedge q(x)\big)$$

$$[\![\mathrm{S}]\!]^{w,g}_{t} = [\![\mathrm{NP}]\!]^{w,g}([\![\mathrm{VP}]\!]^{w,g})$$
$$= \lambda q_{\langle e,t\rangle}.\exists x\big((x \text{ is a person}) \wedge (x \text{ studies language}) \wedge q(x)\big)(\lambda y.y \text{ greeted me})$$
$$= 1 \iff \exists x\big((x \text{ is a person}) \wedge (x \text{ studies language}) \wedge (x \text{ greeted me})\big). \quad \square$$

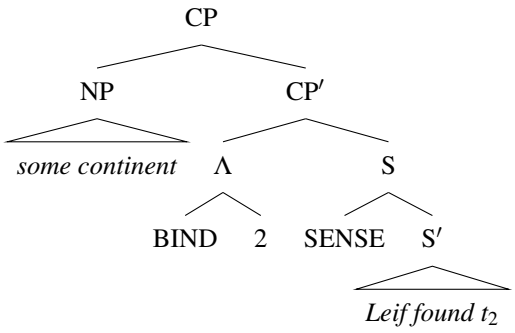
3.2 Quantifier movement

We can adapt this treatment of relative clauses to accommodate moving quantifiers. We will do this by inserting an unpronounced BIND node into the parse tree with the same semantic value as "who".

Definition 3.2 (Denotation of an unpronounced trace binder)**.**

$$[\![\text{BIND}]\!]^g_{\langle z,\langle l,\langle e,t\rangle\rangle\rangle} = [\![who]\!]^g = \lambda i_z.\lambda \alpha_l.\lambda x.[\![\alpha]\!]^{[i\to x]||g}$$

Take, for example, the sentence "Leif found some continent".



$$[\![\text{S}]\!]^{w,g}_l = \text{S}'$$

$$[\![\Lambda]\!]^{w,g}_{\langle l,\langle e,t\rangle\rangle} = [\![\text{BIND}]\!]^{w,g}([\![2]\!]^{w,g}) = (\lambda i_z.\lambda \alpha_l.\lambda x.[\![\alpha]\!]^{[i\to x]||g})(2)$$

$$= \lambda \alpha_l.\lambda x.[\![\alpha]\!]^{[2\to x]||g}$$

$$[\![\text{CP}']\!]^{w,g}_{\langle e,t\rangle} = [\![\Lambda]\!]^{w,g}([\![\text{S}]\!]^{w,g}) = (\lambda \alpha_l.\lambda x.[\![\alpha]\!]^{[2\to x]||g})(\text{S}') = \lambda x.[\![\text{S}']\!]^{[2\to x]||g}$$

$$= \lambda x.\text{Leif found } ([2 \to x]||g)(2) = \lambda x.\text{Leif found } x$$

$$[\![\text{CP}]\!]^{w,g}_t = [\![\text{NP}]\!]^{w,g}([\![\text{CP}']\!]^{w,g})$$

$$= \big(\lambda p_{\langle e,t\rangle}.\exists y(p(y) = 1 \wedge y \text{ is a continent})\big)(\lambda x.\text{Leif found } x)$$

$$= 1 \iff \exists y\big((\lambda x.\text{Leif found } x)(y) = 1 \wedge y \text{ is a continent}\big)$$

$$\iff \exists y(\text{Leif found } y \wedge y \text{ is a continent}). \quad \square$$

4 Alternative semantics

We will use sense to work with alternative semantics by defining a function $\mathfrak{F}$ that recursively computes the set of alternative LF-subtrees for a node α [4, 5]. Because $\mathfrak{F}$ modifies syntactic tree structure directly, we do not have to worry about defining an alternative-sensitive version of every compositional rule in our system.

Definition 4.1 (The alternative set)**.** *Let $\mathfrak{F}(\alpha)$ be the alternative set of α.*

$$\mathfrak{F}(\alpha) = \begin{cases} \text{some contextually determined set of nodes} & f = 1 \\ \{\alpha\} & f = 0 \wedge \text{leaf}(\alpha) \\ \{(\sigma, f, \beta', \gamma') : \beta' \in \mathfrak{F}(\beta) \wedge \gamma' \in \mathfrak{F}(\gamma)\} & f = 0 \wedge \neg\text{leaf}(\alpha) \end{cases}$$

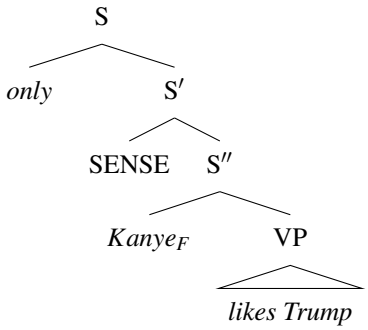
4.1 Non-constituent "only"

To show how this sense formalism interacts with alternative semantics, we will give a denotation for "only" that takes an LF-node α as an argument. Intuitively, a statement like "Only Kanye likes Trump" presupposes that Kanye likes Trump, and asserts that no contextually-determined alternatives to Kanye like Trump [6]. We can formalize this denotation with the alternative set defined in (4.1).

Definition 4.2 (Denotation of non-constituent "only")**.**

$$[\![only]\!]^w_{\langle l,t\rangle} = \lambda\, \alpha_l . \forall \phi_l \in \mathfrak{F}(\alpha)(\phi \neq \alpha \rightarrow [\![\phi]\!]^w = 0)$$

In the derivation below, italic names correspond to leaf nodes, and non-italic names represent individuals. Assume that $\mathfrak{F}(Kanye_F) = \{Kanye, JayZ, Nas\}$.



$$[\![\mathrm{S}']\!]^w_l = \mathrm{S}''$$

$$[\![\mathrm{S}]\!]^w_t = [\![only]\!]^w([\![\mathrm{S}']\!]^w) = \big(\lambda\, \alpha_l . \forall \phi_l \in \mathfrak{F}(\alpha)(\phi \neq \alpha \rightarrow [\![\phi]\!]^w = 0)\big)(\mathrm{S}'')$$

$$= 1 \iff \forall \phi_l \in \mathfrak{F}(\mathrm{S}'')(\phi \neq \mathrm{S}'' \to [\![\phi]\!]^w = 0)$$

$$\mathfrak{F}(\mathrm{S}'') = \{(S'', 0, \beta, \gamma) : \beta \in \mathfrak{F}(Kanye_F) \wedge \gamma \in \mathfrak{F}(\mathrm{VP})\}$$
$$= \{(S'', 0, \beta, \gamma) : \beta \in \{Kanye, JayZ, Nas\} \wedge \gamma \in \{\mathrm{VP}\}\}$$
$$= \{(S'', 0, Kanye, \mathrm{VP}), (S'', 0, JayZ, \mathrm{VP}), (S'', 0, Nas, \mathrm{VP})\}$$

$$\therefore [\![\mathrm{S}]\!]^w = 1 \iff ([\![(S'', 0, JayZ, \mathrm{VP})]\!]_t^w = 0) \wedge ([\![(S'', 0, Nas, \mathrm{VP})]\!]_t^w = 0)$$
$$\iff \big([\![\mathrm{VP}]\!]^w([\![JayZ]\!]^w) = 0\big) \wedge \big([\![\mathrm{VP}]\!]^w([\![Nas]\!]^w) = 0\big)$$
$$\iff \big((\lambda x.x \text{ likes Trump})(\text{Jay-Z}) = 0\big) \wedge \big((\lambda x.x \text{ likes Trump})(\text{Nas}) = 0\big)$$
$$\iff (\text{Jay-Z likes Trump} = 0) \wedge (\text{Nas likes Trump} = 0). \quad \square$$

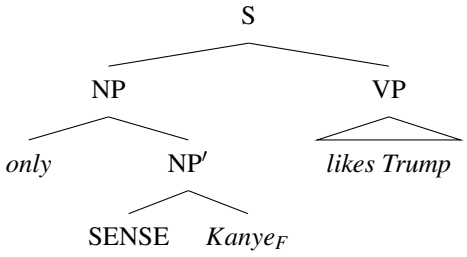
4.2 Generalized "only"

Let $(\lambda_1\ \lambda_2)$ represent the application of one lambda term to the other in whichever order is possible. We will also define the commutative operation $+$ over types, which returns the type of $(\lambda_1\ \lambda_2)$ where λ_1 and λ_2 have the types of the arguments of $+$. Using this notation, we can give a generalized denotation for "only" that handles both constituent and non-constituent usages.

Definition 4.3 (Generalized denotation of "only")**.** [2]

$$[\![only]\!]^w_{\langle l, \langle \tau, t\rangle\rangle} = \lambda\, \alpha_l.\lambda o_\tau : (\tau + \mathrm{type}([\![\alpha]\!]^w) = t).\forall x_l \in \mathfrak{F}(\alpha)\big(x \neq \alpha \to ([\![x]\!]^w\ o) = 0\big)$$

We can now re-derive the truth conditions for the same sentence assuming "only" forms a constituent with "Kanye".



$$[\![\mathrm{NP}']\!]^w_l = Kanye_F$$

$$[\![\mathrm{NP}]\!]^w_{\langle \tau, t\rangle} = [\![only]\!]^w([\![\mathrm{NP}']\!]^w)$$

[2] In fact, the earlier denotation of non-constituent "only" given in (4.2) is a special case of (4.3) when $\mathrm{type}([\![\alpha]\!]^w) = t$ and $o_\tau = \mathrm{Id}$. A proof of this fact is emitted.

$$= \big(\lambda\alpha_l.\lambda o_\tau : (\tau + \text{type}([\![\alpha]\!]^w) = t).\forall x_l \in \mathfrak{F}(\alpha)\big(x \neq \alpha \rightarrow ([\![x]\!]^w\ o) = 0\big)\big)(\mathit{Kanye}_F)$$

$$= \lambda o_\tau : (\tau + e = t).\forall x_l \in \mathfrak{F}(\mathit{Kanye}_F)\big(x \neq \mathit{Kanye}_F \rightarrow ([\![x]\!]^w\ o) = 0\big)$$

$$= \lambda o_\tau : (\tau + e = t).\big(([\![\mathit{JayZ}]\!]^w\ o) = 0\big) \wedge \big(([\![\mathit{Nas}]\!]^w\ o) = 0\big)$$

$$= \lambda o_\tau : (\tau + e = t).\big((\text{Jay-Z}\ o) = 0\big) \wedge \big((\text{Nas}\ o) = 0\big)$$

$$[\![\text{S}]\!]^w_t = [\![\mathit{only}]\!]^w([\![\text{VP}]\!]^w)$$

$$= \big(\lambda o_\tau : (\tau + e = t).\big((\text{Jay-Z}\ o) = 0\big) \wedge \big((\text{Nas}\ o) = 0\big)\big)(\lambda y.y \text{ likes Trump})$$

$$= 1 \iff \big((\lambda y.y \text{ likes Trump})(\text{Jay-Z}) = 0\big) \wedge \big((\lambda y.y \text{ likes Trump})(\text{Nas}) = 0\big)$$

$$\iff (\text{Jay-Z likes Trump} = 0) \wedge (\text{Nas likes Trump} = 0). \quad \square$$

5 Conclusion

The framework of sense abstraction formalizes the notion of sense developed by Frege. It gives us a straightforward method of deriving the truth conditions of intensional statements. In addition, we can use it to handle binding in relative constructions without a special-case compositional rule like predicate abstraction. Finally, it allows us to give denotations for words like "only" that interact with the alternative denotations of other constituents. Thus, sense abstraction provides a unified system for handling intensionality, binding, and alternative semantics without the need for additional compositional rules.

References

[1] Heim, I., & Kratzer, A. (1998). *Semantics in generative grammar* (Vol. 13). Oxford: Blackwell.

[2] Frege, G. (1948). Sense and reference. *The philosophical review*, 57(3), 209-230.

[3] Fintel, K. V., & Heim, I. (2011). Lecture notes in intensional semantics, Ms.

[4] Rooth, M. (1985). Association with focus.

[5] Rooth, M. (1992). A theory of focus interpretation. *Natural language semantics, 1*(1), 75-116.

[6] Horn, L. (1969). A presuppositional approach to only and even. In 5th Regional Meeting, *Chicago Linguistic Society* (pp. 98-107).