0 INTRODUCTION

The primary goal of linguistics can be described as an attempt to understand what a person knows when they know a language. In the context of syntax, this can be seen as an attempt to understand what a speaker of a language has to know about the structure of utterances in order for those utterances to be judged acceptable.

0.1 The Problem

Traditional, theories of sentence construction, focus entirely on the mechanisms by which the parts of a sentence combine. In the Minimalist Program, and its close cousins Combinatory Categorial Grammar and Lexical Tree Adjoining Grammar, this is taken to its logical extreme, with some versions of Minimalism pushing the entirety of the generative mechanism onto combinatory properties of individual syntactic objects, often analogized to combinatory properties of atoms and molecules in chemistry. Even in less extreme theories, however, such as Head-driven Phrase Structure Grammar or Lexical-Functional Grammar, the theory is built entirely out of rules governing how syntactic objects can join together into larger structures.

A significant problem with this approach is that licensing of these structures is left completely unaddressed. This problem is often left as the responsibility of the syntax-semantics interface, but if one takes the view that syntax is an interface, namely between meaning and utterance, than it should arguably be part of the task of syntax to incorporate a means of licensing appropriate syntactic structures for the semantic content of an utterance. For example, reflexive pronouns are licensed in Minimalism strictly by structural
locations, but the inclusion of the correct pronouns in the numeration (or their selection from the lexicon) entirely precedes the point in the derivation of the sentence where they’re known to be acceptable. One alternative to this is some sort of generic syntactic pronoun object that only gets spelled out in the appropriate way once the sentence as a whole has been built, but this merely puts the problem off to a separate part of the generative process without actually explaining it.

0.2 An Alternative

One potential alternative to this, as taken up here by Pattern-driven Phrase Structure Grammar (PPSG), would be to embed the generative system into a larger cognitive system. This is taken to mean that generation of a sentence can be understood as a task-completion process\(^1\) essentially identical to general cognitive task-complete processes. What distinguishes the syntactic process, however, is the means by which the task is completed, and the exact nature of the task, rather than any specific mechanism, making this model very minimal in terms of what grammar actually is. The primary analogy used here is that phrases and words are syntactic tasks used to encode particular parts of the semantic content of the utterance, with sub-phrases being triggered for exactly the same reasons as sub-tasks in a more general process. This is analogy is especially useful because it accords both with the idea from linguistics that meanings of utterances can, at least in part, be compositional (i.e. built from the meanings of the parts), and also with the fact that the Broca’s Area, the part of the brain most often associated with grammatical competence, is a derivative of the

---

\(^1\) By process I do in fact mean a collection of ordered steps. This is analogous to the Minimalist Program’s derivational versions. I do not know whether such an approach is necessary — computationally speaking, it is well established that there is nothing that an atemporal declarative system such as Prolog can do that a temporally ordered procedural system such as C can do, and vice versa. However I do believe that the nature of the human brain, as an object engaging in temporally ordered procedural actions, makes the declarative approach unnecessarily complicated.
motor cortex, which is heavily involved in turning abstract intentions into low-level plans of action for completing motor tasks.

A further aspect of the PPSG model to be outlined here is that the process occurs in a top-down, left-edge generation process. What this means is that the highest phrase structures in the tree representation are identified and used before lower ones, and the left-most substructures are completed first. This top-down, left-edge generation makes it possible to use the existing structure to guide the production of later structures. As I will show in section 2, this also allows for a sort of unification of a number of processes that traditionally are taken to be independent, some integral to the generative process, and others secondary.

0.3 Structure of this Paper

This paper is broken into three primary sections. Section 1 is an overview of the design of PPSG, introducing the idea of semantic graphs, the morphosyntactic structures that instantiate those in the utterance being built, and the general algorithm at work in the generation process. The section also overviews a phenomena called redundancy avoidance. Section 2 looks at the phenomena of head movement and how it is handled in PPSG. Section 3 looks briefly at some differences in lexicalization patterns in English and Romance languages and how they might emerge from historical processes. Section 4 discusses an account of argument structure of conditionally transitivized English verbs like sneeze. Three appendices are provided at the end of the paper: the first contains a number of major morphosyntactic structures for English, the second contains a more detailed mathematical outline of the generative algorithm, and the third contains a brief discussion of the nature of semantic graphs and their relationship to other forms of meaning representations.
1 THE THEORY

The two primary objects operative in the theory proposed here are the semantic graph, which represents the conceptual content conveyed by an utterance, and the morphosyntactic structure, which encodes parts of semantic graphs as morphemes, words, and phrases.

1.1 Semantic Graphs

A semantic graph can be seen as roughly analogous to a collection of simple propositions in a truth conditional representation of meaning. Like a proposition, where there are predicates and their arguments, a semantic graph has two fundamentally different kinds of parts: entities, which represent events, things, and possibly even concepts, and are the nodes in the graph, and relations, which connect entities together, and are the edges in the graph. The analogy between semantic graphs and truth conditions, however, should be taken lightly. Semantic graphs are intended to represent a cognitive construal of the world, or, in a sense, how the person thinks of things as being, as discussed generally in Pinker (2007) (see Gropen, Pinker, Hollander, & Goldberg (1991) for experimental evidence for the role of construals in syntax). A discussion of this, which goes to a greater depth, is provided in Appendix III.

---

2 Semantic graphs are assumed to be directed, typed-edge graphs. One possible way of denoting these is as a set of triples like (edge-type, from-node, to-node). However, for the sake of familiarity, this paper uses the notation edge-type(from-node, to-node), paralleling predication notation. This notation is really only used for in-text reference to particular edges, as whole subgraphs will be given their own graphic.
The easiest way, I think, to understand how semantic graphs work, is to look at a few. Consider first the graph that is encoded by the simple nominal “the cats”:

Here, \( x \) stands in for the thing being mentioned, namely a group of cats. The edge from \( x \) to \( \text{thing} \), labeled \( \text{kind} \), which we can denote here in text as \( \text{kind}(x, \text{thing}) \), indicates that \( x \) is being conceptualized as a thing in the world, as opposed to an event, property, time, etc. The edge \( \text{isa}(x, \text{cat}) \) indicates that the entity is recognized as best being described with the mental category of \( \text{cat} \). The subgraph containing \( \text{person} \), \( \text{number} \), and \( \text{oldnew} \) all indicate some amount of non-semantic content — person is a deictic notion, and could be refined perhaps using one of three deictic concepts, \( \text{self} \) for the speaker, \( \text{addressee} \), and \( \text{other} \); number, like gender, is partially conceptual and partially dependent on the mental category involved, and a more advanced form of semantic graphs might incorporate some sort of complex mental category entity instead of just \( \text{cat} \) to handle this more elegantly; and old and new are in large part dependent on various pragmatic issues such as prior mention.

---

3 By convention, entities are denoted with single letters, \( e_1, e_2, \ldots \) for events, \( x, y, \ldots \) or \( x_1, x_2, \ldots \) for things, \( p, q, \ldots \) or \( p_1, p_2, \ldots \) for properties, \( d, g, \ldots \) or \( d_1, d_2, \ldots \) for degrees, and so forth. The exact letters used really don’t matter, and everything could be indexed \( x \)'s since these aren’t typed variables, but for visual ease of reference the convention here is used.
For now, for the sake of simplicity and lack of a more adequate theory, these less semantic notions will be treated as essentially the same as the rest.

Let’s take a look at a slightly more complicated semantic graph, this time for the phrase “a small dog”:

This graph has two entities, not one — one for the dog, $x$, and one for the dog’s size, $p$. Unlike traditional truth conditional analyses, properties of entities here are assumed to be entities in their own right, in part because they are often capable of being discussed in their own right. However, such an analysis is not essential, and it’s just as easy to represent the dog’s smallness as a relation between the dog entity $x$ and conceptual entity $\text{smallness}$:
Attempting to use such a representation might be possible, but handling notions such as degree is difficult. Traditionally, things like degree are handled with dyadic property predicates as in $\text{small}(x, d)$, where the $d$ variable indicates the degree to which $x$ is small. This paper will attempt to stick to previous notation, for simplicity, and also partially for the sake of elegance — it reduces the number of edge types to a very small set that looks like it would be a good candidate for universal conceptual primitives akin to Pinker (1989) (which is not to say that the members assumed here are themselves good candidates, but you have to start somewhere).

One last graph, now, this time representing the simple sentence “dogs bit the man”: 
The decision to use graphs in this theory is partly a matter of practicality, in that graphs are easier to compute with than traditional truth conditions, and also a matter of theoretical constraint, in that graphs provide a limit to the kinds of things that can be represented. For example, graph edges have precisely two edges, and therefore can only represent dyadic/binary relations between entities. It might turn out that dyadic relations are not sufficient, and p-adic relations need to be substituted (forcing the use of a hypergraph instead of a graph). Similarly, it might turn out that the graph-like structures proposed here are unacceptable, requiring more complicated graphs, or even non-graphs (more discussion on the nature of these graphs can be found in Appendix III). I view this as a benefit of the theory — it provides falsifiable principles, as any good scientific theory should.
1.2 Morphosyntactic Structures

A morphosyntactic structure is very similar to the traditional notion of phrase structure, in that it is a sequence of terminal symbols (phonological strings) and non-terminal symbols (phrase or substructures). However, unlike a phrase structure rule, a morphosyntactic structure is assumed to encodes a particular “property” (in some sense) about some entity that the structure is being used to represent. This property is represented in PPSG by a semantic graph called the “trigger”, and can be any sort of semantic graph that can be used in graph pattern matching (hence “pattern” in the name). This association of meaning with form is reminiscent of the lexical conceptual structures of Jackendoff (1983) and Rappaport & Levin (1988). Alongside the trigger is the phonological structure, which is a list containing any number of phonological strings, as well as collections of ranked substructures that can be recursed into if necessary. This is in contrast to Jackendoff’s structures, and to similar meaning-form pairings such as those found in Construction Grammar, which can have complex partial trees associated with meanings. The set of structures that a person knows is what makes up a persons lexicon, leaving the grammar-proper to consist of the rules of production that operate on these structures.

The notion of morphosyntactic structures given here is essentially free, in that because the idea is derivative on the idea of a more general task, there is nothing unique to the grammar in the nature of the structures. Even the ranked substructures is is essentially free, as it obviously necessary for the cognitive architecture as a whole to be able to choose among alternative subtasks by ranking them according to most relevant to the larger task at hand. Because of this, PPSG arguably very little-m-minimalist.
1.2.1 Triggers

The trigger for a morphosyntactic structure is a semantic graph with variable nodes. That is to say, some of the nodes in the graph are variables that can be bound during a pattern matching process. For example, we might have a morphosyntactic structure √the, which is the lexical root for the English definite determiner the. This structure would have the phonological structure [“the”] triggered by the semantic graph consisting of the edge oldnew(REF, old). A word beginning with uppercase, like REF, is used here to denote a variable. This particular variable, REF, is a special variable reserved for the entity in the graph that the phrase describes. The nature of this variable will be clearer in the following subsection.

At the point during the generation of the sentence when a determiner root is needed in order to encode the definiteness of an entity, the triggers for all the possible determiner roots are tested against the graph. For example, the graph might be:

![Diagram of the graph](image)

If the graph is being used to encode x, then REF is equal to x during the pattern matching phase, so the trigger graph’s sole edge, oldnew(REF, old), matches the graph edge oldnew(x, old), and the phonological structure [“the”] is used for the determiner root.
An important theoretical question that can be asked about the relationship between triggering semantics and morphosyntactic structures is what sort of correspondence between them does natural language permit. It might be the case, for instance, that for any given trigger, there can be only one morphosyntactic structure associated with it, and vice versa, which would produce a (more generalized) version of the Uniform \( \theta \)-Assignment Hypothesis (UTAH) of Baker (1988), applicable not just to the normal \( \theta \)-roles of verb argument structure but to all the abstract entity-relating properties. This would be the most limited case, and is probably a good starting point. Alternatively, there might be many structures associated with a given trigger graph, or maybe multiple graphs associated with a given structure, or both. The model of triggers presented here assumes that morphosyntactic structures have exactly one trigger, but extensions of this are trivial. A related question is how complex can graphs be, and are there any primitives. I assume here that graphs can be arbitrarily complex (an admittedly undesirably unrestrained system), but that a number of structures, roughly those of McCawley (1971), form the basis of the verbal projection.

### 1.2.2 Phonological Structures

Phonological structures (PSs) are slightly more complicated objects than triggers. They are used to represent the content of the trigger in a phonological and/or structural way. They consist, in the simplest case, of just a phonological string. In the previous subsection, the determiner root structure \( \sqrt{\text{the}} \) had the phonological structure \( [“\text{the”}] \), which is just a simple word. In the complex case, an item in a phonological structure is an association of some entity in the trigger graph with a partially ordered set\(^4\) (or “poset”) of substructures that can be placed in that location within the phonological structure, and a graph transformation, which

---

\(^4\) A partially ordered set (poset) is a pair \( (E, \leq) \), where \( E \) is a set of elements, and a binary relation \( \leq \) over \( E \) which is reflexive \( (a \leq a) \), antisymmetric \( (a \leq b \land b \leq a \rightarrow a = b) \), and transitive \( (a \leq b \land b \leq c \rightarrow a \leq c) \). Essentially, a poset is a set in which some, but not necessarily all, elements are ordered or ranked relative to others.
reconstrues the meaning of the sentence, that is performed on the graph before encoding the entity as the relevant substructure. Many substructure posets consist of only a single substructure, especially at the phrasal level, where category structures, like $D$, are used to collect various morphological structures into a single place.

In notation for structures, a phonological string is often just represented by an orthographical form inside quotes, as used earlier for the √the phonological structure. Both [“the”] and [/ði/] represent the phonological structure for same morphosyntactic structure, namely √the. The association of entities with substructure posets and transformations are denoted, in the phonological structure, as symbols with a subscript for the entity, and the poset and graph transformation are given elsewhere$^5$, for convenience of notation, however when only a single structure is a member of a poset, the name for that structure will be used directly in the phonological structure, rather than being placed elsewhere. For example, if we want to model the regular past tense verb suffix -ed, we might use the following:

\[
\text{PastV} \quad \text{PS: } [\alpha_{\text{REF}} \text{“-ed”}] \\
\text{Trigger: } \xrightarrow{\text{time}} \alpha_{\text{REF}} \rightarrow \text{past} \\
\alpha_{X}: \quad \sqrt{\text{kick, toss, ...}; \text{ID}}
\]

The trigger for this structure includes the edge $\text{friend}(\text{REF}, X)$, and the $\alpha_X$ in this phonological structure is used to denote the association of the $X$ entity variable in that edge with the poset consisting of just an $OfP$, and with the identity transformation $ID$. When the transformation is omitted, it is presumed to be the identity transformation. The specification of a poset of structures that can be recursed into is essentially subcategorization, but as will be demonstrated later, it is also responsible for various movement phenomena. Another example of a phonological structure would be

$^5$ Usually posets are represented graphically with Hasse diagrams, but in this paper, since the posets are relatively simple, I use just list elements of the poset separated by commas. Order between them is denoted with >.
the TopicP, where the subject of the English sentence is located, given here with some minimal posets:

\[
\text{TopicP} \quad \text{PS:} \quad [\alpha_x \text{TP}_{\text{REF}}] \\
\text{Trigger:} \quad \xrightarrow{\text{top}}
\]

\[
\alpha_x: \quad \text{DP} > \text{DummyItP}
\]

When there are multiple phrases in a poset, as there are here for \(\alpha_x\), the maximal element of all those that match is selected and produced. The DummyItP would match every semantic graph possible, but because it is ordered lower than DP, any time a DP can be produced, one will be. Only when a DP cannot be produced will the DummyItP be the maximal element, and so we would find the dummy it only in sentences without a topic that can be encoded as a DP (as would be the case in a non-raising use of seem).

1.3 Generation Algorithm

The general process of generating a sentence can be described with a relatively simple algorithm. We start with the task of representing some particular entity \(x\) in the semantic graph \(G\) as some morphosyntactic structure \(S\). This is called “Say”, as in, “Say \(G\) as \(S\)”. If the phonological structure of \(S\) is just a simple phonemic string, the result of saying \(G\) as \(S\) is just the phonological structure as is. If there are any substructure recursions (the \(\alpha\)'s, \(\beta\)'s, etc) within the phonological structure of \(S\), then for each of the recursions we look at the candidate substructure posets, taking those candidate substructures whose triggers match the graph \(G'\) (the result of transforming \(G\) by the associated graph transformation), when the REF variable is assumed to be equal to the same entity that this particular substructure is going to encode for. We then rank those matching substructures according to the

---

6 Topicality of subjects in English is another one of those pragmatics-related things like old and new information. Its treatment here is essentially makeshift until an account of information-structure–related syntax can be given.
poset, pulling out the maximal substructure (i.e. the substructure not outranked by any other substructure). This is called “Select”, as in “Select a substructure to encode the entity $REF$ given semantic graph $G'$, from among the possible substructures $Ss$”. The selected substructure is produced in the same fashion as the first step.

More illustrative of the complex case, suppose we encode the semantic graph $G$ as morphosyntactic structure $S$, which has the phonological structure $[\alpha_x \beta_{REF} \gamma_{REF}]$. Assume $REF = e$ and $X = x$, and assume that $\alpha_x$ has two candidate substructures, $A_1$ and $A_2$, ranked $A_1 > A_2$. We stipulate, then, that the $REF$ variable for $A_1$ and $A_2$’s triggers be equal to $x$, because $\alpha_x$ encodes the entity that the $X$ variable is equal to, which is $x$ in this case. If $T[\alpha_x]$ is the transformation associated with $\alpha_x$, $G'$ is the result of transforming the $G$ with that transformation. Supposing both $A_1$ and $A_2$ match $G'$, we choose the one that is not outranked by the other, which is in this case $A_1$, and perform a recursive Say of $x$ in the graph $G'$ as structure $A_1$. Similarly for $\beta_{REF}$. But suppose instead that $\beta_{REF}$ has the candidate substructures $B_1$ and $B_2$ ranked $B_1 > B_2$, but only $B_2$’s trigger matches the appropriate transformed graph $G''$. In such a scenario, there is no other matching substructure that outranks $B_2$, and so $B_2$ is used to express $REF = e$. Again similarly, for $\gamma_{REF}$ we might have candidate substructures $C_1$, $C_2$, and $C_3$ ranked $C_1 > C_2$, and $C_1 > C_3$. However, this time suppose that none of them match the appropriate transformed graph $G'''$, so there is no maximal candidate. In such a case, no recursive Say would be performed, because the absence of a matching candidate means nothing in this position is triggered into use by the contents of $G'''$.

A distillation of the algorithm into a sort general idea is that there are morphosyntactic structures that are used to encode particular semantics, as denoted by their triggers. The forms of these morphosyntactic structures are their phonological structures, which can contain simple phonemic strings, or potential substructures. Which substructure is used in any given instance of a structure depends on which substructures matter, as determined by whether or not they can encode part of the graph, and which of those are more in some sense preferred, as determined by the
ranking of the potential substructures with a poset. Since it is possible for a language to encode some semantic content or other as a modification of the representation of some other related semantic content (e.g. the causative alternation in English where the agentive event of breaking a vase can be viewed as a modification of the representation for a vase breaking spontaneously), transformations of semantic graphs are associated with the recursive step, the simplest case of which is the identity transform $ID$, which changes nothing in the graph.

### 1.4 Example Derivation

At this point it is possible to look at an example derivation. For this I will use the phrase “a small dog” from earlier. For convenience, here is the semantic graph again:

![Semantic Graph]

The relevant structures here, with some irrelevant parts omitted, and some relevant related structures, might be given as the following:

<table>
<thead>
<tr>
<th>DP</th>
<th>PS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>αREF:</td>
<td>$[D_{REF} \alpha_{REF}]$</td>
</tr>
<tr>
<td>$\alpha_{REF}$:</td>
<td>PropP &gt; NP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>PS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{REF}$:</td>
<td>$[\alpha_{REF}]$</td>
</tr>
<tr>
<td>$\sqrt{a}, \sqrt{the}$</td>
<td></td>
</tr>
</tbody>
</table>
\(\sqrt{a}\)  
PS: [“a”]  
Trigger: \(\text{REF} \xrightarrow{\text{oldnew}} \text{new}\)

\(\sqrt{\text{the}}\)  
PS: [“the”]  
Trigger: \(\text{REF} \xrightarrow{\text{oldnew}} \text{old}\)

PropP  
PS: [AdjP \(\alpha_{\text{REF}}\)]  
Trigger: \(\text{REF} \xrightarrow{\text{hasproperty}} \text{P}\)  
\(\alpha_{\text{REF}}\): PropP > NP

AdjP  
PS: [Adj\(_{\text{REF}}\)]

Adj  
PS: [\(\alpha_{\text{REF}}\)]  
\(\alpha_{\text{REF}}\): \(\sqrt{\text{small}}, \sqrt{\text{large}}\)

\(\sqrt{\text{small}}\)  
PS: [“small”]  
Trigger: \(\text{REF} \xrightarrow{\text{isa}} \text{smallness}\)

\(\sqrt{\text{large}}\)  
PS: [“large”]  
Trigger: \(\text{REF} \xrightarrow{\text{isa}} \text{largeness}\)

NP  
PS: [N\(_{\text{REF}}\)]

N  
PS: [\(\alpha_{\text{REF}}\)]  
\(\alpha_{\text{REF}}\): SgN, PlN

SgN  
PS: [\(\alpha_{\text{REF}}\)]  
Trigger: \(\text{REF} \xrightarrow{\text{number}} \text{sg}\)  
\(\alpha_{\text{REF}}\): \(\sqrt{\text{dog}}, \sqrt{\text{cat}}\)
The derivation for the phrase “a small dog” would then look something like this (omitting unnecessary excessive detail, and including the structure names to mirror bracketed notation):

1. Produce $[DP \ DREF \ a_{REF}]$ with $REF = x$.

2. Produce $[D \ a_{REF}]$ with $REF = x$.
   $a_{REF}$ can be $\sqrt{a}$, $\sqrt{the}$. Maximal matching structure: $\sqrt{a}$.

3. Produce $[\sqrt{a} \ “a”]$ with $REF = x$, encoding
   
   $REF \xrightarrow{old\ to\ new} new$

   $\rightarrow [D [\sqrt{a} \ “a”]]$
   $\rightarrow [DP [D [\sqrt{a} \ “a”]] a_{REF}]$

   $a_{REF}$ can be PropP, NP. Maximal matching structure: PropP.
4. Produce $[\text{PropP AdjP} \, \alpha_{\text{REF}}]$ with $\text{REF} = x$, $P = p$, encoding $\text{REF} \xrightarrow{\text{hasproperty}} P$

5. Produce $[\text{AdjP Adj}_{\text{REF}}]$ with $\text{REF} = p$.

6. Produce $[\text{Adj} \, \alpha_{\text{REF}}]$ with $\text{REF} = p$.
   $\alpha_{\text{REF}}$ can be $\sqrt{\text{small}}, \sqrt{\text{large}}$. Maximal matching element: $\sqrt{\text{small}}$.

7. Produce $[\sqrt{\text{small}} \, \text{"small"}]$ with $\text{REF} = p$, encoding

   $\text{REF} \xrightarrow{\text{isa}} \text{smallness} \xrightarrow{} [\text{Adj} \, [\sqrt{\text{small}} \, \text{"small"}]]$
   $\xrightarrow{} [\text{AdjP} \, [\text{Adj} \, [\sqrt{\text{small}} \, \text{"small"}]]]$
   $\xrightarrow{} [\text{PropP} \, [\text{AdjP} \, [\sqrt{\text{small}} \, \text{"small"}]]] \beta_{\text{REF}}$

   $\alpha_{\text{REF}}$ can be $\text{PropP}, \text{NP}$. Maximal matching element: $\text{NP}$.

8. Produce $[\text{NP N}_{\text{REF}}]$ with $\text{REF} = x$.

9. Produce $[\text{N} \, \alpha_{\text{REF}}]$ with $\text{REF} = x$.
   $\alpha_{\text{REF}}$ can be $\text{SgN}, \text{PlN}$. Maximal matching element: $\sqrt{\text{dog}}$.

10. Produce $[\text{SgN} \, \alpha_{\text{REF}}]$ with $\text{REF} = x$, encoding

    $\text{REF} \xrightarrow{\text{number}} \text{sg}$

    $\alpha_{\text{REF}}$ can be $\sqrt{\text{dog}}, \sqrt{\text{cat}}$. Maximal matching element: $\sqrt{\text{dog}}$.

11. Produce $[\sqrt{\text{dog}} \, \text{"dog"}]$ with $\text{REF} = x$, encoding

    $\text{REF} \xrightarrow{\text{isa}} \text{dog}$

    $\xrightarrow{} [\sqrt{\text{dog}} \, \text{"dog"}]$
    $\xrightarrow{} [\text{SgN} \, [\sqrt{\text{dog}} \, \text{"dog"}]]$
    $\xrightarrow{} [\text{N} \, [\text{SgN} \, [\sqrt{\text{dog}} \, \text{"dog"}]]]$
    $\xrightarrow{} [\text{NP} \, [\text{N} \, [\text{SgN} \, [\sqrt{\text{dog}} \, \text{"dog"}]]]]$
    $\xrightarrow{} [\text{PropP} \, [\text{AdjP} \, [\text{Adj} \, [\sqrt{\text{small}} \, \text{"small"}]]]]$
    $\xrightarrow{} [\text{NP} \, [\text{N} \, [\text{SgN} \, [\sqrt{\text{dog}} \, \text{"dog"}]]]]$
1.5 Redundancy Avoidance

Redundancy avoidance is the general principle that you should never say twice what you can get away with saying only once. A trivial form, that is taken in part from contemporary Minimalism, is the notion of movement (and by extension the phenomena normally seen as control), where a phrase would be produced in two places within a sentence, but is only said in one. Pronominal forms could similarly be generated, but would require slightly more complicated mechanisms than are used here. A second form, dominance-based redundancy avoidance, is somewhat different, but computationally more salient in its necessity, and so I address it first.

1.5.1 Dominance-based Redundancy Avoidance

More important than movement, at least in that failure to move redundancy doesn't produce impossible sentences, is the kind of redundancy avoidance that prevents sentences like this: “The dog that bit the man that was bitten by the dog that ... bit the man that was bitten by the dog that bit the man that ...”. This kind of sentence is very closely related to the problems that arise in Minimalism when trying to reconstruct a quantified phrase in an antecedent-contained deletion scenario:

Given: John [knows everyone Mary does [e]]

If we say...

[e] = [knows everyone Mary [e]]

then reconstruction leads to infinite regress:
Both of these problems can be seen as resulting from a very simple error. If you assume that production of a structure is blind to the structures above it, then taking a semantic graph for a sentence and simply using the whole thing to encode one of the entities in the graph will lead to the whole graph being re-encoded every time any entity is encoded. Put another way, if you try to encode entity $x$ using the whole graph, you must also at some point encode entity $y$ using the whole graph, which in turn means you must encode entity $x$ using the whole graph, etc.

A simple solution would be to simply not use the whole graph to encode entities, and that could work, but it’s a difficult solution to define. A more elegant solution would be to fall back on the underlying metaphor for PPSG, namely that phrases (or morphosyntactic structures, in theory-specific parlance) are kinds of tasks, which are broken down into subtasks, all of a general problem-solving or action-performing nature. It seems rather trivial to assume that tasks on occasion have to be broken down into subtasks that could also, by happenstance, be the same problem that the task itself is helping to complete. It also seems rather trivial to assume that a general cognitive algorithm would simply not try to solve those subtasks, on the grounds that it is already working to solve them in a higher, containing task. PPSG takes that very view: a structure $S$ encoding an entity $x$ with bound trigger $T$ will not be generated if it is a descendant structure of another instance of $S$ encoding $x$ with bound trigger $T$. It’s important to include the bound trigger graph because there are many cases where the same structure will be reused to encode different, but similar, parts of a semantic graph, as in the case of two $PropP$’s to encode two different properties of the same entity. Algorithmic details are in Appendix I to avoid unnecessary clutter.
1.5.2 Movement-based Redundancy Avoidance

Movement-based redundancy avoidance is a semigeneralization of the dominance-based redundancy avoidance mechanism. Where the dominance-based method says to not re-encode a phrase inside of another instance of itself, the movement-based method essentially says not to re-encode a phrase inside of any phrase that contains another instance of itself. This is directly inspired by contemporary Minimalism’s Copy Theory of Movement, in which structurally lower copies are deleted at PF for economy’s sake (Chomsky 1992, personal communication). Put another way, if a structure $S$, encoding an entity $x$, is a descendant of a structure $P$, and $P$ contains as a substructure another instance of $S$, encoding $x$. This is identical to the notation of simple c-command from traditional syntax. Algorithmic details are in Appendix I for the same reasons as before.

2 Head Movement

In a traditional account of movement, a lower head or head-like unit raises to adjoin, in some fashion, with a higher head, often some sort of functional category. The iconic example of this in English is the raising of the highest auxiliary verb into the position of the T head when a full T auxiliary is absent, as can be seen in the pairs of sentences:

John will not **have** gone to the store  
John **has** not ____ gone to the store

John will not **be** gone to the store  
John **is** not ____ gone to the store

John will not **have been** going to the store  
John **has** not ____ **been** going to the store

The PPSG account of head movement is based on very simple reasoning: when a child is learning the content of a phonological
structure, and building the partially ordered sets of substructures that go into the phonological structure, he will hear sentences similar in form to those given above and conclude that these auxiliary verbs can be placed in the TP structure just like a T head could. The poset for that position would then include the auxiliary structures as well as the T structure.

\[
\text{TP} \quad \text{PS: } [\alpha_{\text{REF}} \beta_{\text{REF}}] \\
\text{Trigger: } \begin{array}{c}
\text{REF} \quad \text{time} \\
\rightarrow \quad \text{T}
\end{array}
\]

\[
\begin{align*}
\alpha_{\text{REF}}: & \quad \text{Will, Perf, Prog} \\
\beta_{\text{REF}}: & \quad \text{NegP} > \text{PerfP} > \text{ProgP} > \text{AgentP} \\
& \quad > \text{ThemeP} > \text{GoalP}
\end{align*}
\]

Because these candidates for the \(\alpha_{\text{REF}}\) position are never heard by the child in that position at the same time, but instead are heard there in different conditions depending on the content of the semantic graph, the child will learn to associate those conditions with the appearance of the particular element’s appearance, forming rankings of preferred phrases for the \(\alpha_{\text{REF}}\) position. For example, whenever both \text{time}(\text{REF, future}) and \text{aspect}(\text{REF, perf}) are in the graph, \text{Will} (\text{PS: } [“will”]), which is associated specifically with the former, shows up instead of \text{Perf} (\text{PS: } [“have”]), which is associated with the latter. But in the absence of \text{time}(\text{REF, future}), \text{Perf} does show up. So \text{Will} must be ranked higher than \text{Perf}, even though both can show up in the \(\alpha_{\text{REF}}\) position. Similar reasoning follows for the other structures that can show up there.

Once the maximal structure is in place, the versions that would show up in the lower \text{Perf} and \text{Prog} structures would naturally be omitted by the movement-based redundancy avoidance method described above. Having these structures be candidates for the same position as T, rather than adjoined to a T element, is roughly analogous to substitution head movement from some versions of traditional grammars. The internal inflectional structure that normally would be attributed to a affix T head or to an agreement operation in adjunction head movement would be handled in PPSG.
by morphologically oriented substructures within the \textit{Perf} and \textit{Prog} structures.

3 Lexicalization Patterns in English and Romance

A fairly well known typological trend in English is the tendency for (native) English verbs to conflate motion with manner, encoding manner semantics in the verb, while rejecting direction semantics, leaving directions, goals, etc. to be encoded in adverbial modifiers. Conversely, in Romance languages verbs typically conflate motion with direction, and leave manner semantics for adverbial modifiers (Talmy 1985). Luckily English has borrowed enough verbs from Romance languages that a proper parallel example can be given solely in English, with the equivalents in Italian, French, etc. being (almost) word-for-word parallels. Consider the following two sentences:

(1) The boy \textit{rolled down} the hill.
(2) The boy \textit{descended} the hill \textit{rolling}.

Both of these sentences meaning basically the same as “The boy went down the hill rolling”. I have italicized the portions of the sentences that encode manner, and bolded those that encode direction. A good question to ask is how one language or family could come to overwhelmingly prefer a conflation of motion with manner, while another prefers a conflation of motion with direction. The obvious answer, in an a-theoretic sense, is to conceive of some sort of currently or historically productive morphosyntactic process that licenses this phenomena, and in fact that is precisely what seems to be the case.

English can readily convert verbs of manner into verbs of motion-in-that-manner. Similarly Latin at one stage in its history could relatively freely dislocate prepositions of direction adverbials and prefix them onto the verb, accounting, in some sense, for why the English verbs have prepositional arguments while the Romance verbs have direct objects. The case of productive English derivation
is somewhat uninteresting, so I will only look at the case of the Latin structure.

The Latin constructions, unlike the English ones, display not merely a derivation of a new word from an old word, but rather involve a full syntactic alteration. What would have been “scandere ad/de scale” without conflation, would become “ad/de scandere scale” with conflation (ignoring scrambling of the nominal). I take this to mean that a special phrase-level morphosyntactic structure existed along side the normal structure (call it $DirP$) for direction arguments to the verb ($DirP_2$ for lack of a more inspired name). The structure should look something like this:

$$\textbf{DirP}_2 \quad \text{PS: } [P_D V_{\text{REF}} \text{AgentP}_{\text{REF}}]$$

$$\text{Trigger: } \begin{array}{c}
\text{moving} \\
\text{isa} \\
\text{direction} \\
\text{D}
\end{array}$$

The process by which this would produce in Latin the relevant structures goes like this: $DirP_2$ is triggered by the presence of a direction in the semantic graph for the utterance, plus whatever other semantic or pragmatic factors make this structure useful (left out because I have no idea what they might be — I do not think we will be able to find native speakers of Latin to give us any clues!). The phonological structure really has only one possible form, since the posets have only one item each, and the form is just:

$$\textbf{DirP}_2$$

$$\begin{array}{c}
P_D \\
V_{\text{REF}} \\
\text{AgentP}_{\text{REF}}
\end{array}$$
These are then further produced, of course, but the only one of interest is the AgentP, which would presumably ultimately produce inside it a DirP, making the structure look like:

![Diagram](image)

It is relatively obvious from this that two prepositions will be produced denoting the same directional entity $D$, and so movement-based redundancy avoidance kicks in and the lower instance is omitted, here denoted by $tP$:  

25
In due course, the sequence $P_D+V_{REF}$ would have been reanalyzed as a single verb, and the whole $DirP_2$ structure would vanish as a productive structure. The (lower) $DirP$ probably got reanalyzed as a $ThemeP$, coinciding with the reanalysis of the “hill” phrase becoming a theme, which is in undoubtedly why modern Italian can alternate between using a transitive construction with a bare $NP$ for “hill”, and an intransitive construction with a $PP$ for “hill” — “Il ragazzo ha sceso [NP la collina]” vs. “Il ragazzo è sceso [PP dalla collina]”. Worth noting is that, in Italian at least, use of $descendere$, the modern descendent of Latin $de+scandere$, in these sentences (as $descenso$ in place of $sceso$) is unacceptable. This is because, as one would expect, after $de+scandere$ fused to become a single word, the lexical semantics of this particular item drifted (meaning something more like familial descent, rather than mere downward movement).
An interesting phenomena of the English language, as noted in Goldberg (1995), is its ability to use intransitive verbs transitively in particularly situations. The conditions under which this use is licensed are typical said to be when the construction includes a path-expression in addition to the direct object.

(1a) Frank sneezed.
(1b) * Frank sneezed the tissue.
(1c) Frank sneezed the tissue off the table.

(2a) Genie blinked.
(2b) * Genie blinked the tissue.
(2c) Genie blinked the tissue off the table.

The sort of sentence in the third line of these triplets, of the form $S V O \text{Path}$ is understood as meaning something like $S$ moves $O$ in Path by Ving. Accounting for this in PPSG is relatively simple. Assume a semantic graph for this sentence includes the following as a subgraph, simplifying for clarity:

---

7 This sentence is actually acceptable, grammatically at least, with the admittedly odd reading where Frank sneezes and produces a tissue in doing so. This reading is even more acceptable with an indefinite, as in “Frank sneezed a tissue”. And of course the addition of a path-expression like up, as in “Frank sneezed up a tissue” makes this reading completely clear.

8 Goldberg (1995) actually claims this construction to be ungrammatical, but with the appropriate situation, e.g. Genie from I Dream of Genie doing the action, the use of the verb in this way is acceptable. I take this to mean not that the structure itself is not licensed, but rather that the relevant semantic content is difficult to ground in a situation, and that this is the reason that most people find it odd to use blink in this fashion. Asking around, I have found that the sentence becomes completely acceptable with a different path-expression, as in “Genie blinked the tissue away”, which I take to further indicate that the difficulty of getting the intended reading is not a grammatical one.
We can now define a simple rule of verbal morphology (that is, a morphosyntactic structure that can be generated as a substructure of the $V$ structure) that applies in precisely this situation.

**MoveByVing**

**PS:** $[\alpha_E]$  

**Trigger:**

$$e_2 \rightarrow \text{blinked}$$

$$e_1 \rightarrow \text{moving}$$

$\alpha_E$: PastV, PresentV

Because this rule is triggered by a graph that has precisely all three of agent, theme, and path relations, it will never be applicable when the semantic graph contains no path entity, thus generating
the grammatical sentences (1c) and (2c), while avoiding the ungrammatical simple transitive sentences (1b) and (2b).

Without a path entity in the semantic graph, the verb generated to encode $e_1$ would be \textit{move}, or some other relevant verb, and the \textit{means} relation to $e_2$ would induce the production of a \textit{by Ving} construction, or something else to indicate that portion of the graph.

\section*{Conclusion}

To summarize, the theory of syntax embodied by Pattern-driven Phrase Structure Grammar takes as its core metaphor the morphosyntactic structure as encoding task, pairing a meaning, represented as a semantic graph, with a structure that encodes it. Among the ways it can be encoded is as a mixture of phonological strings and members of partially ordered sets of ranked candidate substructures. During each recursion into a substructure, a transformation is performed on the semantic graph to derive a new one, a form of reconstruing the content of the sentence. Processes of redundancy avoidance, both dominance-based and movement-based, reduce a sentence’s repetition of structures.

A number of theoretical and experimental questions arise, namely: What is the minimum complexity we can get away with in triggers? What is the minimum complexity we can get away with in phonological structures? What are the core semantic relationships, and what are the core semantic entities, if any? Given the evidence (Gropen, et al., 1991) that there is some tight, innate association between syntactic position and construal of an entity, what structures are universal, and in what way are they distinct from acquired structures?

There are also a number of interesting potential future research directions that look promising:
Incorporation of Pustejovskyan Event Structure Syntax

Pustejovsky (1991)’s model of even structure accords very nicely with this model, and might make it possible to reduce a number of presently atomic-but-complex notions like *perfectiveness* or *become* to some more generic event structure notions.

Incorporation of Jackendoffian Parts and Boundaries

Jackendoff (1991)’s model of the composition of things in space and events in time, boundedness, individuatedness, etc. would make it possible to abstract away from “number”, and with respect to events, accords nicely with with a Pustejovskyan event structure model.

Negative Triggers

The current version of triggers in morphosyntactic structures specifies only what must be present in order to trigger the usage of a structure. Negative triggers could specify what must *not* be present. The idea follows cleanly from the reasoning that a trigger is essentially the knowledge of when a structure is acceptable semantically. It follows that similarly, there times when a structure is unacceptable, e.g. the verb *cry* in an causal/agentive semantic graph.
APPENDIX I: SOME MAJOR ABSTRACT STRUCTURES OF ENGLISH

An abstract structure is taken to be a structure that does not contain only phonological strings. It is often the case that these structures do not contain any phonological strings, being truly abstract in the purest sense. The following are some first passes at describing some of the major ones implicated in the English sentence, using naming conventions that should be either already familiar to those working in the Minimalist Program, or easily understood despite being new. The ranking of the θ-related structures is assumed to follow the θ-hierarchy (Jackendoff 1990).

**TopicP**  
**PS:** $[\text{DP}_X \text{TP}_{\text{REF}}]$  
**Trigger:** $X \overset{\text{topic}}{\sim}$

**TP**  
**PS:** $[\alpha_{\text{REF}} \beta_{\text{REF}}]$  
**Trigger:** $\overset{\text{time}}{\text{REF}} \rightarrow \text{T}$  
$\alpha_{\text{REF}}$: Will, Perf, Prog  
$\beta_{\text{REF}}$: NegP > PerfP > ProgP > AgentP  
> ThemeP > GoalP

**NegP**  
**PS:** $[\text{Neg}_{\text{REF}} \alpha_{\text{REF}}]$  
**Trigger:** $\overset{\text{polarity}}{\text{REF}} \rightarrow \text{negative}$  
$\alpha_{\text{REF}}$: PerfP > ProgP > AgentP  
> ThemeP > GoalP

**PerfP**  
**PS:** $[\text{Perf}_{\text{REF}} \alpha_{\text{REF}}]$  
**Trigger:** $\overset{\text{perfectness}}{\text{REF}} \rightarrow \text{perfect}$  
$\alpha_{\text{REF}}$: ProgP > AgentP > ThemeP > GoalP
ProgP
PS: \([\text{ProgREF}\; \alpha_{\text{REF}}]\)
Trigger: \(\alpha_{\text{REF}}: \text{AgentP} > \text{ThemeP} > \text{GoalP}\)

AgentP
PS: \([\text{DP}_X\; \text{VREF}\; \alpha_{\text{REF}}]\)
Trigger: \(\alpha_{\text{REF}}: \text{ThemeP} > \text{GoalP}\)

ThemeP
PS: \([\text{DP}_X\; \text{VREF}\; \text{GoalP}_{\text{REF}}]\)
Trigger: \(\alpha_{\text{REF}}: \text{ThemeP} > \text{GoalP}\)

GoalP
PS: \([\text{PP}_X\; \text{VREF}]\)
Trigger: \(\alpha_{\text{REF}}: \text{GoalP} > \text{NP}\)

DP
PS: \([\text{DP}_{\text{REF}}\; \alpha_{\text{REF}}]\)
\(\alpha_{\text{REF}}: \text{PropP} > \text{NP}\)

PropP
PS: \([\text{AdjP}\; \alpha_{\text{REF}}]\)
Trigger: \(\alpha_{\text{REF}}: \text{PropP} > \text{NP}\)

NP
PS: \([\text{N}_{\text{REF}}]\)

PP
PS: \([\text{P}_{\text{REF}}\; \text{NP}_{\text{REF}}]\)
APPENDIX II: A FUNCTIONAL BREAKDOWN

The algorithm for constructing sentences can conveniently be broken down into two primary functions, *Say* and *Select*. In mathematical notation, we can describe these functions like so:

Let $P$ be a poset of structures;

Let $\text{match } P \ G \ R$ be the subposet of structures in $P$ that match the semantic graph $G$ when $\text{REF}$ equals $R$;

Let $\text{max } X$ be the maximal elements of a poset $X$;

Select $R \ C \ P = \text{max (match } P \ G \ R)$. 

Let $\text{REF}$ be the entity that the phrase being produced is “about”, $\text{SG}$ the semantic graph for the utterance as a whole, $S$ the morphosyntactic structure to encode $\text{REF}$ as;

Let $B = \{X = a, Y = b, \ldots \}$ be a set of variable bindings like gotten from matching $\text{trigger}[S]$ against $G$;

Let $\text{phon}[S] = [\alpha_X \beta_Y \ldots]$ be the phonological structure of $S$;

Let $\text{sub}[\alpha_X]$ be the poset of substructures of denoted by $\alpha_X$, $\text{sub}[\beta_Y]$ those denoted by $\beta_Y$, etc;

Let $T[\alpha_X]$ be the graph transformation denoted by $\alpha_X$, $T[\beta_Y]$ those denoted by $\beta_Y$, etc;

Let $S'_X = \text{Select } B_X (T[\alpha_X] \ G) \text{sub}[\alpha_X]$, the substructure to use for $\alpha_X$, etc;

Let $\Lambda = \lambda p. \text{if } p = s_v \text{ then } \text{Say } B_v (T[s_v] \ G) S'_v \text{ else } p$ be the function giving back a phonological string or the result of *Saying* a substructure;

Say $\text{REF } G \ S = \text{map } \text{phon}[S] \ \Lambda$
The dominance-based redundancy avoidance process can be achieved by tracking the production of phrases in the \textit{Say} and \textit{Select} functions. The modifications are simple. Assuming all the same things as in the previous definitions:

Let \( E = \{S, P, \ldots\} \) be the set of bound structures for all the structures that dominate the current structure;

\[
\text{Select } R \subseteq P \subseteq E = \max ((\text{match } P \text{ G } R) - E),
\]

\[
\text{Let } \Lambda = \lambda p. \begin{cases} \text{if } p = s_v \text{ then } \text{Say } B_v (T[s_v] \text{ G }) S'_v (\{S'_v\} \cup E) \text{ else } p; \\ \text{Say } \text{REF } \text{G } S \subseteq E = \text{map phon}[S] \Lambda \\ = [p | \text{Say } B_x (T[\alpha_x] \text{ G }) S'_x \{S'_x\} \cup E) \text{ else } p \text{; }
\end{cases}
\]

The modification to \textit{Select} ensures that in the cases of undesirable recursion, such as relative clauses, the structure is not repeated, thus preventing “the dog that bit the man that was bitten by the dog that ...”, while also not preventing the desirable recursion of \textit{PropP}'s, etc. as in “the big red dog”.

Similarly, the modifications to the \textit{Say} and \textit{Select} functions required for the movement-based redundancy avoidance process are also relatively simple. Again assuming all the same things as the previous definitions:

\[
\text{Let } \Lambda = \lambda p. \begin{cases} \text{if } p = s_v \land \exists s \in E(S'_v \in \text{phon}[s]) \text{ then } \varepsilon \text{ else if } p = s_v \text{ then } \text{Say } B_v (T[s_v] \text{ G }) S'_v (\{S'_v\} \cup E) \text{ else } p; 
\end{cases}
\]
Say REF G S E = map phon[S] A

= [p | ε | Say B_X (T[α_X] G) S'_X (S'_X \cup E)
  p | ε | Say B_Y (T[β_Y] G) S'_Y (S'_Y \cup E)
  ...].
An alternate notation for semantic graphs could be constructed along the lines of an attribute-value matrix (AVM).

(1a)

(1b)

\[
\begin{bmatrix}
  x \text{ (thing)} \\
  \text{isa} & \text{dog} \\
  \text{oldnew} & \text{new} \\
  \text{person} & \text{3p} \\
  \text{number} & \text{sg} \\
  \text{hasproperty} & \begin{bmatrix} p \text{ (property)} \\
    \text{isa} & \text{smallness} \end{bmatrix}
\end{bmatrix}
\]
All three of these are equivalent. The two AVM versions of the graph differ only in which of the two entities, $x$ or $p$, is the outermost. Because the hasproperty relation is normally denoted as being from the thing to the property, the AVM in (1c) denotes this edge from $p$ to $x$ as $\text{hasproperty}^R$, the $R$ standing of course for “reverse”. One unappealing property of the AVM notation, which is not an issue for the graph notation, is cyclic graphs. To compensate, we can denote the relevant entities with their names, instead of with AVMs, as done for the following abstract graph:

(2a)
What Semantic Graphs Represent

An issue worth discussing briefly is the nature of the semantic graphs used in this theory. As already noted in the section on semantic graphs, I do not intend them to be quite equivalent to truth conditions. They are supposed to be an alternative formulation of the meaningful content of an utterance, roughly analogous to Jackendoff’s conceptual structures. It is not entirely wrote to view them as truth conditions, but one has to be careful when doing so. Truth conditional semantics, because it focuses just on truth conditions, does not disambiguate between what we might call “logic in the representation” and “logic in the content”. That is to say that there is a distinct clear distinction between the logical relationships that an utterance conveys and the logical relationships used to convey.

A good way to think of the difference is to envision the content of an utterance to be a little snapshot of the parts of the speakers knowledge. Suppose we are talking about the sentence “John runs”, which might have the (simplified) truth conditions $\text{running}(e) \land \text{agent}(e, \text{John})$. It is certainly the case that the persons knowledge of event $e$ includes the fact that it is an event of running. It is also certainly the case that the persons knowledge of event $e$ includes the fact that the agent of the even is John. But it seems redundant, at best, and absurd, at worst, to say that the persons knowledge of the event contains a third, distinct proposition that is the conjunction of those two propositions. Why should it? The conjunction of the two propositions can be determined on the fly as needed. Thus the conjunction in the truth condition is part of the logic in the representation of the content of the utterance, and not the logic in the content of the utterance.

Whether conjunction exists in the content of utterances is an interesting question, since it is hard to see quite what would be contributed by it. However, it is certainly the case that negation of some form must be present, as must disjunction and quantification of a sort. How logic in the content of an utterance gets encoded in a graph is a good question that I have no answer to right now. It
might even turn out that no logic in the content is necessary, however this seems to me to be an unlikely possibility. It is partially for this reason, however — the distinction between logic in the representation and logic in the content — that I chose graphs in the first place as the representation of meaning in this theory. A second reason being computational complexity; manipulating truth conditions is far more complicated than manipulating graphs, so there are matters of theoretical simplicity involved as well.

A third, as previously noted in the section that introduced semantic graphs, is that graphs by their very nature are very limited in the kinds of relations they can represent. If relations are edges, then graphs can only encode strictly dyadic relations, with the ability to simulate monadic predicates by using edges starting and ending on the same node, akin to a dyadic predicate like $\text{ref}(x,x)$. This could have real implications for what sorts of linguistic meaning is possible, or what can be encoded by a morphosyntactic structure, and provides a sort of absolute minimum of representational power. This is in comparison to the graph-like representations of ordinary truth conditions of Sowa (1992)'s conceptual graphs, which have whole graphs as nodes of larger graphs (logical operators), quantifier variable binding edges crossing from higher nodes to nodes in lower graphs-as-nodes, etc.

A remaining question that needs to be addressed is the relationship between semantic graphs and the notion of truth. If semantic graphs are not representations of truth conditions, and furthermore, if they represent how a person has construed things as being, then how can a sentence convey propositional content? To address this, I want to step back from the issue of propositional content of whole sentences a bit and look first at the more general issue of the content of any phrase. The normal understanding from truth conditional semantics is that phrases contribute part of truth conditions. In the sentence “the small dog barked”, the phrase “the small dog” contributes something like $\text{small}(x) \land \text{dog}(x)$, as part of the larger truth conditions for the sentence which might be $\text{small}(x) \land \text{dog}(x) \land \text{agent}(e,x) \land \text{barking}(e)$. But this is a very odd notion, given that in isolation it is without a doubt the case that “a small
“dog” denotes an entity, namely, a small dog. Words and phrases seem to denote entities, not truth conditions. Even sentences can be seen as denoting entities, these entities just happen to be events or states or whatnot.

Connecting this back to the notion of truth, I would say that part of the issue can be solved by introducing a secondary set of edges on top of the existing edges in a semantic graph, which would denote things like the speakers propositional attitude towards the referent entity of the utterance. Does the speaker believe this event is a real event, or is the speaker saying this event is not real and does not exist? Does the speaker doubt the realness of the event? Etcetera. This brings up the problem of just what entities are, if we can talk about entities that the speaker does not believe exist. The simplest answer, the one that I think is most appropriate, is that entities are mental constructs (the idea of an event like so and so). Bridging the obvious gap between mental and real, of course, has been a problem since time immemorial, and I leave that to philosophical discussions. Suffice it to say, all that I feel is necessary for explanatory purposes is to say that utterances are judged true or false by comparing the construal content of an utterance to a construal of the world.
REFERENCES


