An Algebra for Semantic Construction in Constraint-based Grammars

Ann Copestake
Computer Laboratory
University of Cambridge
New Museums Site
Pembroke St, Cambridge, UK
aac@cl.cam.ac.uk

Alex Lascarides
Division of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, Scotland, UK
alex@cogsci.ed.ac.uk

Dan Flickinger
CSLI, Stanford University *and*
YY Software
Ventura Hall, 220 Panama St
Stanford, CA 94305, USA
danf@csli.stanford.edu

Abstract

We develop a framework for formalizing semantic construction within grammars expressed in typed feature structure logics, including HPSG. The approach provides an alternative to the lambda calculus; it maintains much of the desirable flexibility of unification-based approaches to composition, while constraining the allowable operations in order to capture basic generalizations and improve maintainability.

1 Introduction

Some constraint-based grammar formalisms incorporate both syntactic and semantic representations within the same structure. For instance, Figure 1 shows representations of typed feature structures (TFSs) for *Kim*, *sleeps* and the phrase *Kim sleeps*, in an HPSG-like representation, loosely based on Sag and Wasow (1999). The semantic representation expressed is intended to be equivalent to $r\_name(x, Kim) \wedge sleep(e, x)$.[1] Note:

1. Variable equivalence is represented by coindexation within a TFS.
2. The coindexation in *Kim sleeps* is achieved as an effect of instantiating the SUBJ slot in the sign for *sleeps*.
3. Structures representing individual predicate applications (henceforth, elementary predications, or EPs) are accumulated by an append operation. Conjunction of EPs is implicit.
4. All signs have an index functioning somewhat like a λ-variable.

A similar approach has been used in a large number of implemented grammars (see Shieber (1986) for a fairly early example). It is in many ways easier to work with than λ-calculus based approaches (which we discuss further below) and has the great advantage of allowing generalizations about the syntax-semantics interface to be easily expressed. But there are problems. The operations are only specified in terms of the TFS logic: the interpretation relies on an intuitive correspondence with a conventional logical representation, but this is not spelled out. Furthermore the operations on the semantics are not tightly specified or constrained. For instance, although HPSG has the Semantics Principle (Pollard and Sag, 1994) this does not stop the composition process accessing arbitrary pieces of structure, so it is often not easy to conceptually disentangle the syntax and semantics in an HPSG. Nothing guarantees that the grammar is *monotonic*, by which we mean that in each rule application the semantic content of each daughter subsumes some portion of the semantic content of the mother (i.e., no semantic information is dropped during composition): this makes it impossible to guarantee that certain generation algorithms will work effectively. Finally, from a theoretical perspective, it seems clear that substantive generalizations are being missed.

Minimal Recursion Semantics (MRS: Copestake et al (1999), see also Egg (1998)) tightens up the specification of composition a little. It enforces monotonic accumulation of EPs by making all rules append the EPs of their daughters (an approach which was followed by Sag and Wasow (1999)) but it does not fully spec-

[1] The variables are free, we will discuss scopal relationships and quantifiers below.

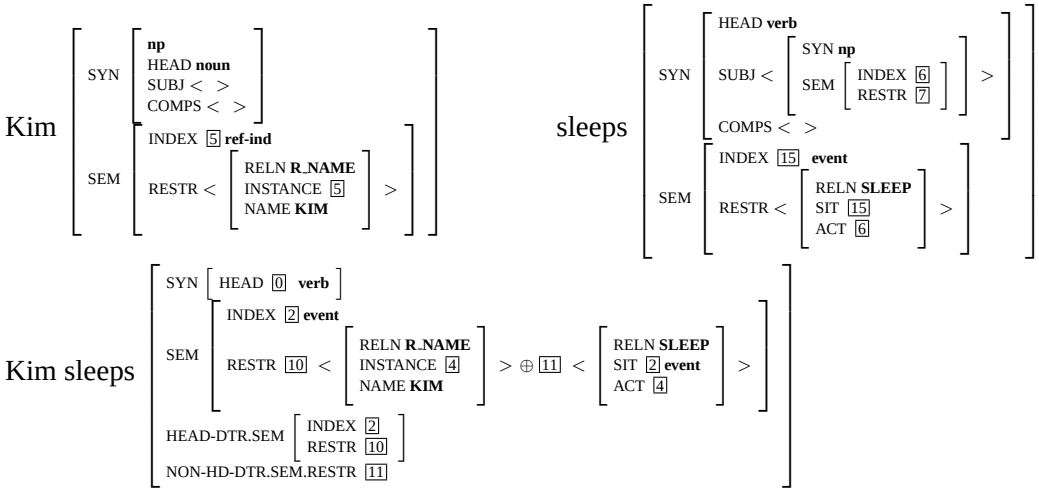

Figure 1: Expressing semantics in TFSs

ify compositional principles and does not formalize composition. We attempt to rectify these problems, by developing an algebra which gives a general way of expressing composition. The semantic algebra lets us specify the allowable operations in a less cumbersome notation than TFSs and abstracts away from the specific feature architecture used in individual grammars, but the essential features of the algebra can be encoded in the hierarchy of lexical and constructional type constraints. Our work actually started as an attempt at rational reconstruction of semantic composition in the large grammar implemented by the LinGO project at CSLI (available via `http://lingo.stanford.edu`). Semantics and the syntax/semantics interface have accounted for approximately nine-tenths of the development time of the English Resource Grammar (ERG), largely because the account of semantics within HPSG is so underdetermined.

In this paper, we begin by giving a formal account of a very simplified form of the algebra and in §3, we consider its interpretation. In §4 to §6, we generalize to the full algebra needed to capture the use of MRS in the LinGO English Resource Grammar (ERG). Finally we conclude with some comparisons to the λ-calculus and to other work on unification based grammar.

2 A simple semantic algebra

The following shows the equivalents of the structures in Figure 1 in our algebra:

Kim: $[x_2]\{[]_{subj}, []_{comp}\}[r\_name(x_2, Kim)]\{\}$
sleeps: $[e_1]\{[x_1]_{subj}, []_{comp}\}[sleep(e_1, x_1)]\{\}$
Kim sleeps: $[e_1]\{[]_{subj}, []_{comp}\}[sleep(e_1, x_1), r\_name(x_2, Kim)]\{x_1 = x_2\}$

The last structure is semantically equivalent to: $[sleep(e_1, x_1), r\_name(x_1, Kim)]$.

In the structure for *sleeps*, the first part, $[e_1]$, is a *hook* and the second part ($[x_1]_{subj}$ and $[]_{comp}$) is the *hole*s. The third element (the *lzt*) is a bag of elementary predications (EPs).[2] Intuitively, the hook is a record of the value in the semantic entity that can be used to fill a hole in another entity during composition. The holes record gaps in the semantic form which occur because it represents a syntactically unsaturated structure. Some structures have no holes, such as that for *Kim*. When structures are composed, a hole in one structure (the semantic head) is filled with the hook of the other (by equating the variables) and their lzts are appended. It should be intuitively obvious that there is a straightforward relationship between this algebra and the TFSs shown in Figure 1, although there are other TFS architectures which would share the same encoding.

We now give a formal description of the algebra. In this section, we simplify by assuming that each entity has only one hole, which is unlabelled, and only consider two sorts of variables: events and individuals. The set of semantic entities is built from the following vocabulary:

[2] As usual in MRS, this is a bag rather than a set because we do not want to have to check for/disallow repeated EPs; e.g., *big big car*.

1. The absurdity symbol $\perp$.
2. indices $i_1, i_2, \ldots$, consisting of two subtypes of indices: events $e_1, e_2, \ldots$ and individuals $x_1, x_2, \ldots$.
3. n-place predicates, which take indices as arguments
4. $=$.

Equality can only be used to identify variables of compatible sorts: e.g., $x_1 = x_2$ is well formed, but $e = x$ is not. Sort compatibility corresponds to unifiability in the TFS logic.

Definition 1 *Simple Elementary Predications (*SEP*)*
An SEP *contains two components:*

1. *A relation symbol*
2. *A list of zero or more ordinary variable arguments of the relation (i.e., indices)*

This is written relation(arg$_1, \ldots,$arg$_n$). For instance, like(e, x, y) is a well-formed SEP.

Equality Conditions: Where i_1 and i_2 are indices, $i_1 = i_2$ is an equality condition.

Definition 2 *The Set Σ of Simple semantic Entities (*SSEMENT*)*
$s \in \Sigma$ if and only if $s = \perp$ or $s = \langle s_1, s_2, s_3, s_4 \rangle$ such that:

- *$s_1 = \{[i]\}$ is a hook;*
- *$s_2 = \emptyset$ or $\{[i']\}$ is a hole;*
- *s_3 is a bag of* SEP*s(the lzt)*
- *s_4 is a set of equalities between variables (the eqs).*

We write a SSEMENT as: $[i_1][i_2][\text{SEP}s]\{\text{EQ}s\}$. Note for convenience we omit the set markers $\{\}$ from the hook and hole when there is no possible confusion. The SEPs, and EQs are (partial) *descriptions* of the fully specified formulae of first order logic.

Definition 3 The Semantic Algebra
A Semantic Algebra defined on vocabulary V is the algebra $\langle \Sigma, op \rangle$ where:

- *Σ is the set of* SSEMENT*s defined on the vocabulary V, as given above;*
- *$op : \Sigma \times \Sigma \longrightarrow \Sigma$ is the operation of semantic composition. It satisfies the following conditions. If $a_1 = \perp$ or $a_2 = \perp$ or* hole$(a_2) = \emptyset$*, then $op(a_1, a_2) = \perp$. Otherwise:*
  1. hook$(op(a_1, a_2))$ = hook(a_2)
  2. hole$(op(a_1, a_2))$ = hole(a_1)
  3. lzt$(op(a_1, a_2))$ = lzt$(a_1) \oplus$ lzt(a_2)
  4. eq$(op(a_1, a_2)) = Tr(eq(a_1) \cup eq(a_2) \cup$ hook(a_1) = hole$(a_2)\})$
     where Tr stands for transitive closure *(i.e., if $S = \{x = y, y = z\}$, then $Tr(S) = \{x = y, y = z, x = z\}$).*

This definition makes a_2 the equivalent of a semantic functor and a_1 its argument.

Theorem 1 *op is a function*
If $a_1 = a_3$ and $a_2 = a_4$, then $a_5 = op(a_1, a_2) = op(a_3, a_4) = a_6$. Thus op is a function. Furthermore, the range of op is within Σ. So $\langle \Sigma, op \rangle$ is an algebra.

We can assume that semantic composition always involves two arguments, since we can define composition in ternary rules etc as a sequence of binary operations. Grammar rules (i.e., constructions) may contribute semantic information, but we assume that this information obeys all the same constraints as the semantics for a sign, so in effect such a rule is semantically equivalent to having null elements in the grammar. The correspondence between the order of the arguments to op and linear order is specified by syntax.

We use variables and equality statements to achieve the same effect as coindexation in TFSs. This raises one problem, which is the need to avoid accidental variable equivalences (e.g., accidentally using x in both the signs for *cat* and *dog* when building the logical form of *A dog chased a cat*). We avoid this by adopting a convention that each instance of a lexical sign comes from a set of basic *sement*s that have pairwise distinct variables. The equivalent of coindexation within a lexical sign is represented by repeating the same variable but the equivalent of coindexation that occurs during semantic composition is an equality condition which identifies two different variables. Stating this formally is straightforward but a little long-winded, so we omit it here.

3 Interpretation

The SEPs and EQs can be interpreted with respect to a first order model $\langle E, A, F\rangle$ where:

1. E is a set of events
2. A is a set of individuals
3. F is an interpretation function, which assigns tuples of appropriate kinds to the predicates of the language.

The **truth definition** of the SEPs and EQs (which we group together under the term SMRS, for simple MRS) is as follows:

1. For all events and individuals v, $[\![v]\!]^{\langle M,g\rangle} = g(v)$.
2. For all n-predicates P^n, $[\![P^n]\!]^{\langle M,g\rangle} = \{\langle t_1,\ldots,t_n\rangle : \langle t_1,\ldots,t_n\rangle \in F(P^n)\}$.
3. $[\![P^n(v_1,\ldots,v_n)]\!]^{\langle M,g\rangle} = 1$ iff $\langle [\![v_1]\!]^{\langle M,g\rangle},\ldots,[\![v_n]\!]^{\langle M,g\rangle}\rangle \in [\![P^n]\!]^{\langle M,g\rangle}$.
4. $[\![\phi \wedge \psi]\!]^{\langle M,g\rangle} = 1$ iff $[\![\phi]\!]^{\langle M,g\rangle} = 1$ and $[\![\psi]\!]^{\langle M,g\rangle} = 1$.

Thus, with respect to a model M, an SMRS can be viewed as denoting an element of $\mathcal{P}(G)$, where G is the set of variable assignment functions (i.e., elements of G assign the variables $e,\ldots$ and $x,\ldots$ their denotations):

$$[\![smrs]\!]^M = \{g : g \text{ is a variable assignment function and } M \models_g smrs\}$$

We now consider the semantics of the algebra. This must define the semantics of the operation op in terms of a function f which is defined entirely in terms of the denotations of op's arguments. In other words, $[\![op(a_1,a_2)]\!] = f([\![a_1]\!],[\![a_2]\!])$ for some function f. Intuitively, where the SMRS of the SEMENT a_1 denotes G_1 and the SMRS of the SEMENT a_2 denotes G_2, we want the semantic value of the SMRS of $op(a_1,a_2)$ to denote the following:

$$G_1 \cap G_2 \cap [\![hook(a_1) = hole(a_2)]\!]$$

But this *cannot* be constructed purely as a function of G_1 and G_2.

The solution is to add hooks and holes to the denotations of SEMENTS (cf. Zeevat, 1989). We define the denotation of a SEMENT to be an element of $I \times I \times \mathcal{P}(G)$, where $I = E \cup A$, as follows:

Definition 4 *Denotations of* SEMENTs
If $a \neq \bot$ is a SEMENT, $[\![a]\!]^M = \langle [i],[i'],G\rangle$ *where:*

1. $[i] = \mathit{hook}(a)$
2. $[i'] = \mathit{hole}(a)$
3. $G = \{g : M \models_g \mathit{smrs}(a)\}$

$[\![\bot]\!]^M = \langle \emptyset,\emptyset,\emptyset\rangle$

So, the meanings of SEMENTS are ordered three-tuples, consisting of the hook and hole elements (from I) and a set of variable assignment functions that satisfy the SMRS.

We can now define the following operation f over these denotations to create an algebra:

Definition 5 *Semantics of the Semantic Construction Algebra*
$\langle I \times I \times \mathcal{P}(G), f\rangle$ is an algebra, where:

$f(\langle\emptyset,\emptyset,\emptyset\rangle,\langle[i_2],[i'_2],G_2\rangle) = \langle\emptyset,\emptyset,\emptyset\rangle$
$f(\langle[i_1],[i'_1],G_1\rangle,\langle\emptyset,\emptyset,\emptyset\rangle) = \langle\emptyset,\emptyset,\emptyset\rangle$
$f(\langle[i_1],[i'_1],G_1\rangle,\langle[i_2],\emptyset,G_2\rangle = \langle\emptyset,\emptyset,\emptyset\rangle$
$f(\langle[i_1],[i'_1],G_1\rangle,\langle[i_2],[i'_2],G_2\rangle) =$
$\quad\langle[i_2],[i'_1],G_1 \cap G_2 \cap G'\rangle$
where $G' = \{g : g(i_1) = g(i'_2)\}$

And this operation demonstrates that semantic construction is compositional:

Theorem 2 *Semantics of Semantic Construction is Compositional*
The mapping $[\![]\!] : \langle\Sigma, op\rangle \longrightarrow \langle\langle I,I,G\rangle,f\rangle$ is a homomorphism (so $[\![op(a_1,a_2)]\!] = f([\![a_1]\!],[\![a_2]\!])$).

This follows from the definitions of **[]**, op and f.

4 Labelling holes

We now start considering the elaborations necessary for real grammars. As we suggested earlier, it is necessary to have multiple labelled holes. There will be a fixed inventory of labels for any grammar framework, although there may be some differences between variants.[3] In HPSG, complements are represented using a list, but in general there will be a fixed upper limit for the number of complements so we can label holes COMP1, COMP2, etc. The full inventory of labels for

[3] For instance, Sag and Wasow (1999) omit the distinction between SPR and SUBJ that is often made in other HPSGs.

the ERG is: SUBJ, SPR, SPEC, COMP1, COMP2, COMP3 and MOD (see Pollard and Sag, 1994).

To illustrate the way the formalization goes with multiple slots, consider op_{subj}:

Definition 6 The definition of op_{subj}
$op_{subj}(a_1, a_2)$ *is the following: If* $a_1 = \perp$ *or* $a_2 = \perp$ *or* $\text{hole}_{subj}(a_2) = \emptyset$*, then* $op_{subj}(a_1, a_2) = \perp$*. And if* $\exists l \neq subj$ *such that:*

$$|\text{hole}_l(a_1) \cup \textit{hole}_l(a_2)| > 1$$

then $op_{subj}(a_1, a_2) = \perp$*. Otherwise:*

1. $\textit{hook}(op_{subj}(a_1, a_2)) = \textit{hook}(a_2)$
2. *For all labels* $l \neq subj$*:* $\textit{hole}_l(op_{subj}(a_1, a_2)) = \textit{hole}_l(a_1) \cup \textit{hole}_l(a_2)$
3. $\textit{lzt}(op_{subj}(a_1, a_2)) = \textit{lzt}(a_1) \oplus \textit{lzt}(a_2)$
4. $\textit{eq}(op_{subj}(a_1, a_2)) = Tr(eq(a_1) \cup eq(a_2) \cup \{\textit{hook}(a_1) = \textit{hole}_{subj}(a_2)\})$
   where Tr *stands for transitive closure.*

There will be similar operations op_{comp1}, op_{comp2} etc for each labelled hole. These operations can be proved to form an algebra $\langle \Sigma, op_{subj}, op_{comp1}, \ldots \rangle$ in a similar way to the unlabelled case shown in Theorem 1. A little more work is needed to prove that op_l is closed on Σ. In particular, with respect to clause 2 of the above definition, it is necessary to prove that $op_l(a_1, a_2) = \perp$ or for all labels l', $|\textit{hole}_{l'}(op_l(a_1, a_2))| \leq 1$, but it is straightforward to see this is the case.

These operations can be extended in a straightforward way to handle simple constituent coordination of the kind that is currently dealt with in the ERG (e.g., *Kim sleeps and talks* and *Kim and Sandy sleep*); such cases involve daughters with non-empty holes of the same label, and the semantic operation equates these holes in the mother SEMENT.

5 Scopal relationships

The algebra with labelled holes is sufficient to deal with simple grammars, such as that in Sag and Wasow (1999), but to deal with scope, more is needed. It is now usual in constraint based grammars to allow for underspecification of quantifier scope by giving labels to pieces of semantic information and stating constraints between the labels. In MRS, labels called *handles* are associated with each EP. Scopal relationships are represented by EPs with handle-taking arguments. If all handle arguments are filled by handles labelling EPs, the structure is fully scoped, but in general the relationship is not directly specified in a logical form but is constrained by the grammar via additional conditions (handle constraints or *hcons*).[4] A variety of different types of condition are possible, and the algebra developed here is neutral between them, so we will simply use rel_h to stand for such a constraint, intending it to be neutral between, for instance, $=_q$ (qeq: equality modulo quantifiers) relationships used in MRS and the more usual $\leq$ relationships from UDRT (Reyle, 1993). The conditions in hcons are accumulated by append.

To accommodate scoping in the algebra, we will make hooks and holes *pairs* of indices and handles. The handle in the hook corresponds to the LTOP feature in MRS. The new vocabulary is:

1. The absurdity symbol $\perp$.
2. handles $h_1, h_2, \ldots$
3. indices $i_1, i_2, \ldots$, as before
4. n-predicates which take handles and indices as arguments
5. rel_h and $=$.

The revised definition of an EP is as in MRS:

Definition 7 *Elementary Predications (*EPs*)*
An EP *contains exactly four components:*

1. *a handle, which is the label of the* EP
2. *a relation*
3. *a list of zero or more ordinary variable arguments of the relation (i.e., indices)*
4. *a list of zero or more handles corresponding to scopal arguments of the relation.*

[4] The underspecified scoped forms which correspond to sentences can be related to first order models of the fully scoped forms (i.e., to models of WFFs without labels) via supervaluation (e.g., Reyle, 1993). This corresponds to stipulating that an underspecified logical form u entails a base, fully specified form ϕ only if all possible ways of resolving the underspecification in u entails ϕ. For reasons of space, we do not give details here, but note that this is entirely consistent with treating semantics in terms of a description of a logical formula. The relationship between the SEMENTS of non-sentential constituents and a more 'standard' formal language such as λ-calculus will be explored in future work.

This is written h:r($a_1, \ldots, a_n$,$sa_1, \ldots, sa_m$). For instance, h:every(x, h_1, h_2) is an EP.[5]

We revise the definition of semantic entities to add the hcons conditions and to make hooks and holes pairs of handles and indices.

H-Cons Conditions: Where h_1 and h_2 are handles, $h_1 rel_h h_2$ is an H-Cons condition.

Definition 8 *The Set Σ of Semantic Entities $s \in \Sigma$ if and only if $s = \bot$ or $s = \langle s_1, s_2, s_3, s_4, s_5 \rangle$ such that:*

- $s_1 = \{[h, i]\}$ *is a hook;*
- $s_2 = \emptyset$ *or* $\{[h', i']\}$ *is a hole;*
- s_3 *is a bag of* EP *conditions*
- s_4 *is a bag of* HCONS *conditions*
- s_5 *is a set of equalities between variables.*

SEMENTs are: $[h_1, i_1]\{holes\}[eps][hcons]\{eqs\}$.

We will not repeat the full composition definition, since it is unchanged from that in §2 apart from the addition of the append operation on hcons and a slight complication of eq to deal with the handle/index pairs:
$eq(op(a_1, a_2)) = Tr(eq(a_1) \cup eq(a_2) \cup$
$\{hdle(hook(a_1)) = hdle(hole(a_2)),$
$ind(hook(a_1)) = ind(hole(a_2))\})$
where Tr stands for *transitive closure* as before and *hdle* and *ind* access the handle and index of a pair. We can extend this to include (several) labelled holes and operations, as before. And these revised operations still form an algebra.

The truth definition for SEMENTS is analogous to before. We add to the model a set of labels L (handles denote these via g) and a well-founded partial order $\leq$ on L (this helps interpret the *hcons*; cf. Fernando (1997)). A SEMENT then denotes an element of $\mathcal{H} \times \ldots \mathcal{H} \times \mathcal{P}(G)$, where the $\mathcal{H}$s ($= L \times I$) are the new hook and holes.

Note that the language Σ is first order, and we do not use λ-abstraction over higher order elements.[6] For example, in the standard Montagovian view, a quantifier such as *every* is represented by the higher-order expression $\lambda P \lambda Q \forall x(P(x), Q(x))$. In our framework, however, *every* is the following (using qeq conditions, as in the LinGO ERG):

$[h_f, x]\{[]_{subj}, []_{comp1}, [h', x]_{spec}, \ldots\}$
$[h_e : \textit{every}(x, h_r, h_s)][h_r =_q h']\{\}$

and *dog* is:

$[h_d, y]\{[]_{subj}, []_{comp1}, []_{spec}, \ldots\}[h_d : \textit{dog}(y)][]\{\}$

So these composes via op_{spec} to yield *every dog*:

$[h_f, x]\{[]_{subj}, []_{comp1}, []_{spec}, \ldots\}$
$[h_e : \textit{every}(x, h_r, h_s), h_d : \textit{dog}(y)]$
$[h_r =_q h']\{h' = h_d, x = y\}$

This SEMENT is semantically equivalent to:

$[h_f, x]\{[]_{subj}, []_{comp1}, []_{spec}, \ldots\}$
$[h_e : \textit{every}(x, h_r, h_s), h_d : \textit{dog}(x)][h_r =_q h_d]\{\}$

A slight complication is that the determiner is also syntactically selected by the N′ via the SPR slot (following Pollard and Sag (1994)). However, from the standpoint of the compositional semantics, the determiner is the semantic head, and it is only its SPEC hole which is involved: the N′ must be treated as having an empty SPR hole.

In the ERG, the distinction between intersective and scopal modification arises because of distinctions in representation at the lexical level. The repetition of variables in the SEMENT of a lexical sign (corresponding to TFS coindexation) and the choice of type on those variables determines the type of modification.

Intersective modification: *white dog*:

dog: $[h_d, y]\{[]_{subj}, []_{comp1}, \ldots, []_{mod}\}$
$[h_d : \textit{dog}(y)][]\{\}$

white: $[h_w, x]\{[]_{subj}, []_{comp1}, .., [h_w, x]_{mod}\}$
$[h_w : \textit{white}(x)][]\{\}$

white dog: $[h_w, x]\{[]_{subj}, []_{comp1}, \ldots, []_{mod}\}$
(op_{mod}) $[h_d : \textit{dog}(y), h_w : \textit{white}(x)][]$
$\{h_w = h_d, x = y\}$

Scopal Modification: *probably walks*:

walks: $[h_w, e']\{[h', x]_{subj}, []_{comp1}, \ldots, []_{mod}\}$
$[h_w : \textit{walks}(e', x)][]\{\}$

probably: $[h_p, e]\{[]_{subj}, []_{comp1}, \ldots, [h, e]_{mod}\}$
$[h_p : \textit{probably}(h_s)][h_s =_q h]\{\}$

probably walks: $[h_p, e]\{[h', x]_{subj}, []_{comp1}, \ldots, []_{mod}\}$
(op_{mod}) $[h_p{:}\textit{probably}(h_s), h_w{:}\textit{walks}(e', x)]$
$[h_s =_q h]\{h_w = h, e = e'\}$

[5] Note *every* is a predicate rather than a quantifier in this language, since MRSs are partial descriptions of logical forms in a base language.

[6] Even though we do not use λ-calculus for composition, we could make use of λ-abstraction as a representation device, for instance for dealing with adjectives such as *former*, cf., Moore (1989).

6 Control and external arguments

We need to make one further extension to allow for control, which we do by adding an extra slot to the hooks and holes corresponding to the external argument (e.g., the external argument of a verb always corresponds to its subject position). We illustrate this by showing two uses of *expect*; note the third slot in the hooks and holes for the external argument of each entity. In both cases, x'_e is both the external argument of *expect* and its subject's index, but in the first structure x'_e is also the external argument of the complement, thus giving the control effect.

expect 1 (as in *Kim expected to sleep*)
$[h_e, e_e, x'_e]\{[h_s, x'_e, x'_s]_{subj}, [h_c, e_c, x'_e]_{comp1}, \ldots\}$
$[h_e : \textit{expect}(e_e, x'_e, h'_e)][h'_e =_q h_c]\{\}$

expect 2 (*Kim expected that Sandy would sleep*)
$[h_e, e_e, x'_e]\{[h_s, x'_e, x'_s]_{subj}, [h_c, e_c, x'_c]_{comp1}, \ldots\}$
$[h : \textit{expect}(e_e, x'_e, h'_e)][h'_e =_q h_c]\{\}$

Although these uses require different lexical entries, the semantic predicate *expect* used in the two examples is the same, in contrast to Montagovian approaches, which either relate two distinct predicates via meaning postulates, or require an additional semantic combinator. The HPSG account does not involve such additional machinery, but its formal underpinnings have been unclear: in this algebra, it can be seen that the desired result arises as a consequence of the restrictions on variable assignments imposed by the equalities.

This completes our sketch of the algebra necessary to encode semantic composition in the ERG. We have constrained accessibility by enumerating the possible labels for holes and by stipulating the contents of the hooks. We believe that the handle, index, external argument triple constitutes all the semantic information that a sign should make accessible to a functor. The fact that only these pieces of information are visible means, for instance, that it is impossible to define a verb that controls the object of its complement.[7] Although obviously changes to the syntactic valence features would necessitate modification of the hole labels, we think it unlikely that we will need to increase the inventory further. In combination with the principles defined in Copestake et al (1999) for qeq conditions, the algebra presented here results in a much more tightly specified approach to semantic composition than that in Pollard and Sag (1994).

[7] Readers familiar with MRS will notice that the KEY feature used for semantic selection violates these accessibility conditions, but in the current framework, KEY can be replaced by KEYPRED which points to the predicate alone.

7 Comparison

Compared with λ-calculus, the approach to composition adopted in constraint-based grammars and formalized here has considerable advantages in terms of simplicity. The standard Montague grammar approach requires that arguments be presented in a fixed order, and that they be strictly typed, which leads to unnecessary multiplication of predicates which then have to be interrelated by meaning postulates (e.g., the two uses of *expect* mentioned earlier). Type raising also adds to the complexity. As standardly presented, λ-calculus does not constrain grammars to be monotonic, and does not control accessibility, since the variable of the functor that is λ-abstracted over may be arbitrarily deeply embedded inside a λ-expression.

None of the previous work on unification-based approaches to semantics has considered constraints on composition in the way we have presented. In fact, Nerbonne (1995) explicitly advocates nonmonotonicity. Moore (1989) is also concerned with formalizing existing practice in unification grammars (see also Alshawi, 1992), though he assumes Prolog-style unification, rather than TFSs. Moore attempts to formalize his approach in the logic of unification, but it is not clear this is entirely successful. He has to divorce the interpretation of the expressions from the notion of truth with respect to the model, which is much like treating the semantics as a description of a logic formula. Our strategy for formalization is closest to that adopted in Unification Categorial Grammar (Zeevat et al, 1987), but rather than composing actual logical forms we compose partial *descriptions* to handle semantic underspecification.

8 Conclusions and future work

We have developed a framework for formally specifying semantics within constraint-based representations which allows semantic operations in

a grammar to be tightly specified and which allows a representation of semantic content which is largely independent of the feature structure architecture of the syntactic representation. HPSGs can be written which encode much of the algebra described here as constraints on types in the grammar, thus ensuring that the grammar is consistent with the rules on composition. There are some aspects which cannot be encoded within currently implemented TFS formalisms because they involve negative conditions: for instance, we could not write TFS constraints that absolutely prevent a grammar writer sneaking in a disallowed coindexation by specifying a path into the lzt. There is the option of moving to a more general TFS logic but this would require very considerable research to develop reasonable tractability. Since the constraints need not be checked at runtime, it seems better to regard them as metalevel conditions on the description of the grammar, which can anyway easily be checked by code which converts the TFS into the algebraic representation.

Because the ERG is large and complex, we have not yet fully completed the exercise of retrospectively implementing the constraints throughout. However, much of the work has been done and the process revealed many bugs in the grammar, which demonstrates the potential for enhanced maintainability. We have modified the grammar to be monotonic, which is important for the chart generator described in Carroll et al (1999). A chart generator must determine lexical entries directly from an input logical form: hence it will only work if all instances of nonmonotonicity can be identified in a grammar-specific preparatory step. We have increased the generator's reliability by making the ERG monotonic and we expect further improvements in practical performance once we take full advantage of the restrictions in the grammar to cut down the search space.

Acknowledgements

This research was partially supported by the National Science Foundation, grant number IRI-9612682. Alex Lascarides was supported by an ESRC (UK) research fellowship. We are grateful to Ted Briscoe, Alistair Knott and the anonymous reviewers for their comments on this paper.

References

Alshawi, Hiyan [1992] (ed.) *The Core Language Engine*, MIT Press.

Carroll, John, Ann Copestake, Dan Flickinger and Victor Poznanski [1999] An Efficient Chart Generator for Lexicalist Grammars, *The 7th International Workshop on Natural Language Generation*, 86–95.

Copestake, Ann, Dan Flickinger, Ivan Sag and Carl Pollard [1999] Minimal Recursion Semantics: An Introduction, manuscript at `www-csli.stanford.edu/~aac/newmrs.ps`

Egg, Marcus [1998] *Wh*-Questions in Underspecified Minimal Recursion Semantics, *Journal of Semantics*, 15.1:37–82.

Fernando, Tim [1997] Ambiguity in Changing Contexts, *Linguistics and Philosophy*, 20.6: 575–606.

Moore, Robert C. [1989] Unification-based Semantic Interpretation, *The 27th Annual Meeting for the Association for Computational Linguistics (ACL-89)*, 33–41.

Nerbonne, John [1995] Computational Semantics—Linguistics and Processing, Shalom Lappin (ed.) *Handbook of Contemporary Semantic Theory*, 461–484, Blackwells.

Pollard, Carl and Ivan Sag [1994] *Head-Driven Phrase Structure Grammar*, University of Chicago Press.

Reyle, Uwe [1993] Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction, *Journal of Semantics*, 10.1: 123–179.

Sag, Ivan, and Tom Wasow [1999] *Syntactic Theory: An Introduction*, CSLI Publications.

Shieber, Stuart [1986] *An Introduction to Unification-based Approaches to Grammar*, CSLI Publications.

Zeevat, Henk [1989] A Compositional Approach to Discourse Representation Theory, *Linguistics and Philosophy*, 12.1: 95–131.

Zeevat, Henk, Ewan Klein and Jo Calder [1987] An introduction to unification categorial grammar, Nick Haddock, Ewan Klein and Glyn Morrill (eds), *Categorial grammar, unification grammar, and parsing: working papers in cognitive science, Volume 1*, 195–222, Centre for Cognitive Science, University of Edinburgh.