LG518 Problems with simple Context Free Phrase Structure Grammars

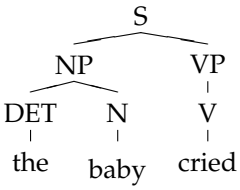
Doug Arnold
University of Essex
doug@essex.ac.uk

1 Introduction

The central idea of (Context Free) Phrase Structure Grammar is to describe the hierarchical structure of phrases by means of rules, which define trees:

(1) a. $S \rightarrow NP\ VP$
b. $VP \rightarrow V\ (NP)\ (PP)$
c. $PP \rightarrow P\ NP$
d. $NP \rightarrow DET\ N$

(2)



It is 'context free' because the choice of rule to expand a non-terminal does not depend on the context. This is a very attractive model of grammar:

- directional neutrality (neutral with respect to bottom-up vs top-down interpretation);
- process neutrality (neutral with respect to generation or parsing)
- it has both a 'generative' (proof theoretic) and an entirely constraint based, static (model theoretic) interpretation.

But there are several problems:

2 Problem: Redundancy, Missed Generalization

In a Context Free PSG, non-terminal symbols are atoms. Thus, while one can write rules like the following, TV_{sing} and TV_{plur} are completely unrelated to each other, and to IV_{sing}.

(3) a. $S \rightarrow NP_{sing}\ VP_{sing}$
b. $S \rightarrow NP_{plur}\ VP_{plur}$
c. $VP_{sing} \rightarrow IV_{sing}$
d. $VP_{sing} \rightarrow TV_{sing}\ NP_{sing}$
e. $VP_{sing} \rightarrow TV_{sing}\ NP_{plur}$
etc., etc.

HPSG Solution: non-terminals are complex, possibly underspecified, categories (attribute-value structures, descriptions of feature structures).

We need a richer notion of 'matching' (to define what it means for a tree to satisfy a rule): the notion of *unification*. See below.

3 Problem: Headedness

There is nothing to prevent linguistically crazy rules such as:

(4) $VP \rightarrow DET\ PP$

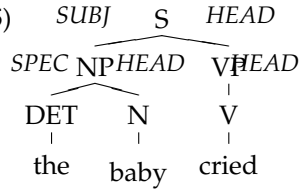
i.e. there is no idea of *headedness* (endocentricity).

HPSG Solution: Notion of Headedness, and the Head Feature Principle (HFP).

In headed constructions, one item is identified as the head. The HFP requires the head and the mother to have certain properties in common (specifically, they must share the value of the HEAD attribute).

This means the branches between S and VP, and VP and V in (5) should be labelled 'HEAD'.

(5)



4 Problem: Complement Selection

There is nothing to rule out:

(6) a. Fido barked.
b. * Fido barked the bone.
c. * Fido enjoyed.
d. Fido enjoyed the bone.

Subcategorization: Heads dictate properties of complements.

5 Formal Foundations

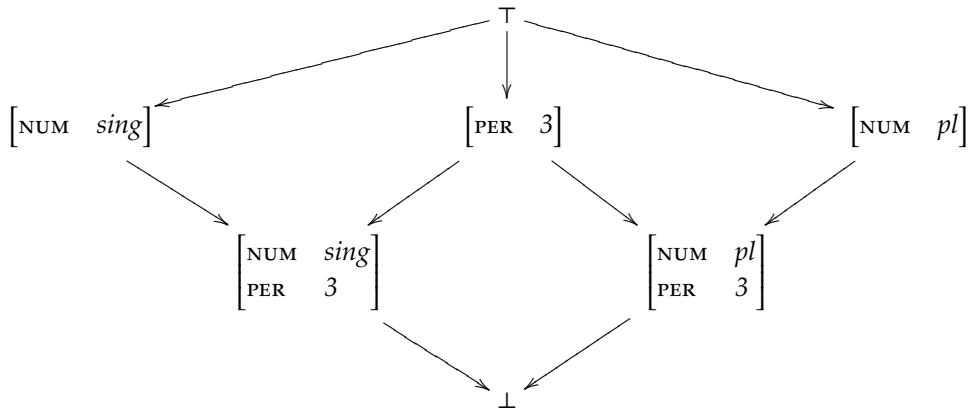
5.1 Subsumption, Unification

There is a very natural partial order on FS, corresponding to increased specificity of descriptions (subsumption).

$A \sqsubseteq B$ A subsumes B (A is more general than B)
$B \sqsupseteq A$ B extends A (B is more specific than A)

This can be pictured as a lattice.

5.2 Subsumption Lattice



Combining (i.e. conjoining) descriptions (i.e. sets of constraints) A and B produces a description that is at least as specific as either: we talk about the *unification* of A and B.

(7) $\left[\textsc{num} \quad sing\right] \sqcup \left[\textsc{per} \quad 3\right] = \begin{bmatrix}\textsc{num} & sing \\ \textsc{per} & 3\end{bmatrix}$

(8) $\left[\textsc{num} \quad sing\right] \sqcup \left[\textsc{num} \quad pl\right] = \bot$ (corresponding to failure of unification)

This can very easily be made precise (the unification of descriptions A and B is the most general description that:

- combines the information in A and B ; or equivalently:
- extends both A and B; or equivalently:
- is subsumed by A and B.

5.3 Type vs Token identity and Re-entrance

Feature structures can be 're-entrant', in the sense that two distinct paths can point to the same object (as opposed to two distinct objects of the same type).

Similarly, in descriptions, different attributes can have the 'same value' in two ways:

(9) token identity: $\begin{bmatrix} \textsc{f} & \boxed{1}\begin{bmatrix}\textsc{h}_1 & v_1 \\ \textsc{h}_2 & v_2\end{bmatrix} \\ \textsc{g} & \boxed{1}\end{bmatrix}$

(10) type identity: $\begin{bmatrix} \textsc{f} & \boxed{1}\begin{bmatrix}\textsc{h}_1 & v_1 \\ \textsc{h}_2 & v_2\end{bmatrix} \\ \textsc{g} & \begin{bmatrix}\textsc{h}_1 & v_1 \\ \textsc{h}_2 & v_2\end{bmatrix}\end{bmatrix}$

Consider what happens if we unify them with (11):

(11) $\left[\textsc{f} \quad \left[\textsc{h}_3 \quad v_3\right]\right]$

Type identity arises 'accidentally' — token identity is what the grammar cares about. (Note: no

'copying' of features: token identity does all the work).

Consider an example like the following, and suppose that NPs carry an agreement attribute of some kind indicating, *inter alia* person, number, gender (and maybe humanness).

(12) Who$_{\boxed{1}}$ did Sandy say $\Delta_{\boxed{1}}$ likes herself$_{\boxed{1}}$?

(Assume we can write rules to ensure that the fronted *wh*-item, the subject of *likes*, and the reflexive have the same value for this agreement feature, as indicated here by $\boxed{1}$).

On encountering *Who*, we know that $\boxed{i}$ must be human; *likes* requires its subject to be 3rd person and singular, and *herself* is feminine.

So we know automatically that $\boxed{1}$ is 3rd, singular, human and feminine, and (e.g.) that the question is about a 3rd singular female human. We can also explain why these are impossible:

(13) a. *What$_{\boxed{1}}$ did Sandy say $\Delta_{\boxed{1}}$ likes herself$_{\boxed{1}}$?
b. *Who$_{\boxed{1}}$ did Sandy say $\Delta_{\boxed{1}}$ like herself$_{\boxed{1}}$?
c. *Who$_{\boxed{1}}$ did Sandy say $\Delta_{\boxed{1}}$ likes themselves$_{\boxed{1}}$?

But can get:

(14) Who$_{\boxed{1}}$ did Sandy say $\Delta_{\boxed{1}}$ like themselves$_{\boxed{1}}$?

5.4 Typed Feature Structures

The Feature Structures used in HPSG are *typed* (or 'sorted').

Each type declares what attributes are appropriate (and perhaps what type of value they can have). For example:

- the attribute CASE is appropriate for nouns, but not verbs.
- (in English) the attribute TENSE is appropriate for verbs, but not adjectives, nouns or determiners.

This gives us a way of expressing co-occurrence restrictions on features (e.g. a word/phrase has TENSE if and only if it is verbal).

Conceptually, typing of FSs allows us to think about when an FS is 'complete' (viz: when every attribute in it has an appropriate value, that value is itself complete), e.g. English nouns are either definite or indefinite (have a value for DEFINITE), and not 'over-' or 'wrongly-specified' (e.g. does not contain inappropriate features, so a *verb* cannot have value for CASE; in English, nouns and adjectives do not have a TENSE feature).

Unification has to respect typing:

(15) $\begin{bmatrix} a & \\ A_1 & v_1 \end{bmatrix} \sqcup \begin{bmatrix} b & \\ A_2 & v_2 \end{bmatrix} = \begin{bmatrix} c & \\ A_1 & v_1 \\ A_2 & v_2 \end{bmatrix}$

(this is only possible if $c = a \sqcup b$, which requires that c is a subtype of both a and b).

Notice that 'atomic' values are just types that have no appropriate attributes.

5.5 Advantages of Unification

A constraint based approach (one that uses unification) has a number of attractions:

- describing things in terms of attributes and values is a very natural way of describing all kinds

of knowledge – it makes part of the language learning task look more like general learning (classification, etc.), and allows us to focus on those aspects of linguistic knowledge (and acquisition) that are really special.

- it gives a precise and explicit notion of 'feature' (a theory of features), e.g. when two feature descriptions are 'the same', when one is more general than another, and gives a precise meaning to ideas like 'underspecification' of categories.
- it makes it easy to express generalizations, e.g. to organize (e.g.) lexical entries into inheritance hierarchies, so that it is easy to state that something is a property of all nouns or all verbs, or that something is a property of all finite verbs, etc.
- it allows us to work with partial information and underspecification (a partial/underspecified description is still a description); this is very important in language processing, because it allows *incremental* interpretation.
- Unification is order independent (commutative, and associative), and a theory that uses it yields grammars that are purely *declarative*. A purely *declarative* grammar is *much easier to understand* than one where the order of operations is significant.
- from a processing point of view, it gives *flexibility* (cf. in example (12), it does not matter in what order the information about the index comes in.
- If grammatical knowledge is purely declarative, it helps us to explain how the same knowledge can be used for different tasks (e.g. comprehension, production) without bias towards any one (cf. trying to 'reverse' a transformational grammar for use in parsing is very difficult).

6 Reading

Sag et al. (2003) Pollard and Sag (1994) Green (2000), Borsley (1996).

References

Robert D. Borsley. *Modern Phrase Structure Grammar*. Number 11 in Blackwell textbooks in linguistics. Blackwell Publishers, 1996.

Georgia Green. Elementary principles of hpsg. URL `http://clwww.essex.ac.uk/papers/hpsg/green.ps`. Unpublished Ms, January 2000.

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

Ivan A. Sag, Thomas Wasow, and Emily M. Bender. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, 2 edition, 2003. URL `http://cslipublications.stanford.edu/site/1575864002.html`.