RTT Grammar Fragment

Andy Lücking

2020

With *Referential Transparency Theory* (RTT), Lücking and Ginzburg (2022) developed a plural semantics for count nouns. RTT provides a witness-based and compositional theory of quantification. Within the RTT article, a few lexical entries and grammatical derivations of sentences are shown. This paper provides a grammar fragment that implements RTT. Hence, it is an appendix to the main article. The grammar fragment is formulated within a type-theoretical 'emulation' of a *Head-driven Phrase Structure Grammar* (HPSG). Some extensions to RTT, most notably collective and distributive interpretations and a replacement of scope in terms of depedent functions, are developed in Lücking (2022).

Contents

1 Introduction

The grammar framework employed here is a variant of *Head-driven Phrase Structure Grammar* (HPSG; Pollard and Sag 1994; Sag, Wasow and Bender 2003), specifically its TTR implementation (HPSG_{TTR}; Cooper 2008; Ginzburg 2012). We use HPSG, because its structured representation format in terms of attribute value matrices allows one to address elements by attribute names. We use HPSG_{TTR}, because the type-theoretical version allows us to directly incorporate the semantic objects introduced in the previous sections.

Our strategy is to pursue a hybrid approach: we implement a TTR analogue to unification in order to capture syntactic agreement and lexical restrictions. Semantic composition is done in terms of function application. In order to streamline grammar representations in an HPSG_{TTR} format, we factor out common features of constructions into general constraints, as is done in

HPSG in terms of *principles* (Sag, Wasow and Bender, 2003). The grammar fragment extends that given in the appendix of Lücking, Ginzburg and Cooper (2021).

2 Grammar fragment

2.1 Merge and the general sign architecture

To start with, the TTR analogue to unification, namely *merge types*, have to be introduced.

(1) a. If R_1 and R_2 are record types, then $R_1 \wedge_{merge} R_2$ is a record type and is called the *merge* of R_1 and R_2.

b. Since merge types are complicated to define (but see Cooper 2012), we follow the strategy of Cooper (2017) and illustrate the working of merges by means of some examples:

(i) $\begin{bmatrix} \text{a} : T \\ \text{b} : R \end{bmatrix} \wedge_{merge} \begin{bmatrix} \text{c} : S \end{bmatrix} = \begin{bmatrix} \text{a} : T \\ \text{b} : R \\ \text{c} : S \end{bmatrix}$

(ii) $\begin{bmatrix} \text{a} : T \end{bmatrix} \wedge_{merge} \begin{bmatrix} \text{a} : R \end{bmatrix} = \begin{bmatrix} \text{a} : T \wedge_{merge} R \end{bmatrix}$

Mainly in order to capture morpho-syntactic information we introduce *tag types* following feature value re-entrances from constraint-based grammars—thereby making the connection transparent and facilitating the readability of HPSG_{TTR} to readers familiar with 'classic' HPSG. we even use the same typographical means, namely boxed numbers. In fact, tag types abbreviate singleton types across different paths and their prime area of use is agreement: the tag type in (2a) is an abbreviation for the structure in (2b) (we use the slash notation in order to indicate paths starting at the outmost level of a feature structure):

(2) a. $\begin{bmatrix} \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{agr} = 3\text{sing}_{\boxed{1}} : Agr \end{bmatrix} \\ \text{spr} : \left\langle \begin{bmatrix} \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{agr} = \boxed{1} : Agr \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$

b. $\begin{bmatrix} \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{agr} = 3\text{sing} : Agr \end{bmatrix} \\ \text{spr} : \left\langle \begin{bmatrix} \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{agr} = /\text{cat.head.agr} : Agr \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$

The agreement type *Agr* is defined in the usual way as follows:

(3) $Agr := \begin{bmatrix} \text{per} : Per(son) \\ \text{num} : Num(erus) \\ \text{gen} : Gend(er) \end{bmatrix}$

A *sign* is a structure that hosts phonetic, syntactic and semantic information:

(4) $$sign := \begin{bmatrix} \text{phon} & : Phoneme \\ \text{cat} & : SynCat \\ \text{dgb-params} & : RecType \\ \text{cont} & : SemObj \end{bmatrix}$$

Signs are distinguished in terms of their constituency status or construction type (*cxtype*) into *lexemes*, *words* and *phrases*. Using the merge operation, constituent types can be represented compactly as constraints over *sign*. For instance, a *word* is defined in (5a), which expands to the structure given in (5b):

(5) a. $word := sign \wedge_{merge} [\text{cxtype} : word] : \text{RecType}$

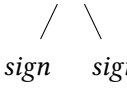
b. $$\begin{bmatrix} \text{cxtype} & : word \\ \text{phon} & : Phoneme \\ \text{cat} & : SynCat \\ \text{dgb-params} & : RecType \\ \text{cont} & : SemObj \end{bmatrix}$$

Words are the output of lexical rules, whose inputs are lexemes:

(6) $lexeme := sign \wedge_{merge} [\text{cxtype} : lexeme] : \text{RecType}$

A phrase is distinguished from words by having daughters. We confine ourselves to phrases with two daughters here, which are represented by means of binary trees as illustrated in (7):

(7) $phrase := sign \wedge_{merge} [\text{cxtype} : phrase]$



If one of the daughters of a phrase chiefly determines (as a default at least) the syntactic properties of the mother, it is the *head* of the phrasal construction, which in turn then is a *headed phrase*. A headed phrase has a daughter labelled as **head** which passes on the syntactic head feature to the mother (indicated by a merge type). The mother also appends the daughters' *phon* values:

(8)
$$\begin{bmatrix} \text{cxtype} & : phrase \\ \text{phon} & : List(\boxed{3}, \boxed{2}) \\ \text{cat} & : \left[\text{head}=\boxed{1} : PoS\right] \\ \text{cont} & : SemObj \end{bmatrix}$$

$$\left[\text{phon} : List(Phoneme)_{\boxed{3}}\right] \quad \textbf{head} := \begin{bmatrix} \text{phon} & : List(Phoneme)_{\boxed{2}} \\ \text{cat} & : \left[\text{head}_{\boxed{1}} : PoS\right] \\ \text{cont} & : SemObj \end{bmatrix}$$

Using constraints and merge types a great deal of linguistic information can be factored out so that lexical entries just have to specify idiosyncratic information of words.

2.2 Lexical entries for nouns, verbs, quantifiers

One upshot of the discussion of the distributive quantifiers in Lücking and Ginzburg (2022, §4.7), is that they involve a syntactic feature that triggers a distributive interpretation of the predicate that combines with the NP whose determiner the quantifier is. To this end, a binary-valued feature *distr* is added to nouns.[1] The raison d'être of *distr* is as follows: *distr*=+ encodes the requirement that a predicate taking the so-marked NP as argument has a distributive interpretation (i.e., $\overrightarrow{PType}^{\text{dist}}$). The marking *distr*=− on the other hand does not *force* a distributive interpretation, although it allows one (i.e., $\overrightarrow{PType}$). Since according to the underlying grammar framework the head of a quantificational NP is the noun instead of the determiner (i.e., an NP instead of a DP analysis is pursued), the plural feature has to be made part of the noun's head feature structure in order to get projected to the phrasal level where it is visible for the verb. Furthermore, since we have argued that singular is just a special case of plural, nouns are lexicalised as pluralities, involving the maxset-refset-compset-triplet right from the beginning. The extended lexical geometry for nouns is as follows:

(9) $noun \mapsto$

$$\begin{bmatrix}
\text{cxtype} & : \textit{lexeme} \\
\text{phon} & : \textit{list}(\textit{Phoneme}) \\
\text{cat} & : \begin{bmatrix}
\text{head} : \begin{bmatrix}
\text{pos=n} : \textit{PoS} \\
\text{agr}\boxed{1} : \begin{bmatrix} \text{num} : \textit{Num} \\ \text{case} : \textit{Case} \\ \text{gend} : \textit{Gend} \end{bmatrix} \\
\text{distr} : \textit{Binary}
\end{bmatrix} \\
\text{spr} : \left\langle \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=det} : \textit{PoS} \\ \text{agr=}\boxed{1} : \textit{AgrType} \\ \text{count} : \textit{Binary} \end{bmatrix} \end{bmatrix} \right\rangle
\end{bmatrix} \\
\text{dgb-params} & : \begin{bmatrix} \text{maxset} & : \textit{Set}(\textit{Ind}) \\ \text{refset} & : \textit{Set}(\textit{Ind}) \\ \text{compset} & : \textit{Set}(\textit{Ind}) \end{bmatrix} \\
\text{cont} & : \textit{SemObj}
\end{bmatrix}$$

[1] Note that for the same reason the feature "count" is part of a determiner's sign structure Sag, Wasow and Bender (2003, p. 112), since verbs do not select for count or mass nouns. Nouns are classified by selecting "count=+" or "count=−" specifiers. Much more syntactic features are used by Beghelli and Stowell (1997) (like "Wh" (marking *Wh*-phrases), "Univ" (indicating universally quantification), or "Neg" (for negation), which are partly captured in purely semantic terms in the present account). For reasons of elegance or systematics, it could be worthwhile to introduce a uniform place for a quantificational syntax-semantics interface for noun phrases (say, a new feature bundle *quantificational type*, or simply "qtype"), which eventually can be extended by further features, if required.

The *semantic object* (*SemObj*) type for a count noun's content will usually be an individual (*Ind*) or a set of individuals (*Set*(*Ind*))—this issue will be of concern for lexical rules below.

The lexeme geometry for intransitive verbs is given in (10):

(10) *verb* $\mapsto$

$$\begin{bmatrix} \text{cxtype} : \textit{lexeme} \\ \text{phon} : \textit{list}(\textit{Phoneme}) \\ \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=v} : \textit{PoS} \\ \text{agr}\boxed{1} : \begin{bmatrix} \text{num} : \textit{Num} \\ \text{pers} : \textit{Pers} \\ \text{gend} : \textit{Gend} \end{bmatrix} \end{bmatrix} \\ \text{spr} : \left\langle \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=n} : \textit{PoS} \\ \text{agr=}\boxed{1} : \textit{AgrType} \\ \text{distr:} \textit{Binary} \end{bmatrix} \end{bmatrix} \right\rangle \\ \text{comps} : \textit{list}(\textit{Sign}) \end{bmatrix} \\ \text{cont} : \textit{SemObj} \end{bmatrix}$$

Both nouns and verbs agree with their specifiers (spr), but the parts of speech of the specifiers differ. It is part of the lexical entry for *every* that it combines with a head noun that is 'distr=+'. The lexical entry for *every* is given in (11):

(11)
$$\begin{bmatrix} \text{phon} : \text{every} \\ \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=det} : \textit{PoS} \\ \text{agr} : \begin{bmatrix} \text{num=sg} : \textit{Num} \end{bmatrix} \\ \text{count=+} : \textit{Binary} \end{bmatrix} \\ \text{spec} : \left\langle \begin{bmatrix} \text{cat} : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=n} : \textit{PoS} \\ \text{distr=+} : \textit{Binary} \end{bmatrix} \end{bmatrix} \\ \text{q-params} : \begin{bmatrix} \text{maxset} : \textit{Set}(\textit{Ind}) \\ \text{refset} : \textit{Set}(\textit{Ind}) \\ \text{compset} : \textit{Set}(\textit{Ind}) \\ \text{c1} : \text{union(refset,compset,maxset)} \end{bmatrix} \end{bmatrix} \right\rangle \\ \text{cont} : \begin{bmatrix} \text{q-cond} : |/\text{cat.spec.q-params.refset}| = |/\text{cat.spec.q-params.maxset}| \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The peculiar thing about *every* is that it semantically applies to a plurality while it syntactically combines with a singular noun. This peculiarity is mostly overlooked or at least not dealt with in quantifier semantics. This might partly be due to the circumstance that this poses non-trivial problems for a compositional grammar, since a non-uniform syntax-semantics interface is required. Since the rich, referentially transparent architecture somewhat blurs a sharp contrast between singular and plural in semantics anyway, we can model *every*-type determiners as follows: they combine with a syntactically singular noun but contributing a plural quantifier condition.

2.3 Lexical rules

Agreement values are instantiated by lexical rules that take lexemes as input and return word forms as output. The lexical rule that produces plural count nouns—that is, nouns whose specifier's value for the *count* feature is '+'—is as follows:

(12) **Plural Rule for Count Nouns (PL-count)**

$$\text{in:}\begin{bmatrix}\text{cxtype} & : \textit{lexeme}\\ \text{phon} & : \textit{list}(\textit{Phoneme})\\ \text{cat} & : \begin{bmatrix}\text{head} : \begin{bmatrix}\text{pos=n} : \textit{PoS}\end{bmatrix}\\ \text{spr} : \left\langle\begin{bmatrix}\text{head} : \begin{bmatrix}\text{count=+} : \textit{Binary}\end{bmatrix}\end{bmatrix}\right\rangle\end{bmatrix}\\ \text{dgb-params} & : \begin{bmatrix}\text{refset} & : \textit{Set}(\textit{Ind})\\ \text{maxset} & : \textit{Set}(\textit{Ind})\\ \text{compset} & : \textit{Set}(\textit{Ind})\\ \text{c1} & : \text{union(refset,compset,maxset)}\\ \text{c2} & : \overrightarrow{\textit{PType}}(\text{maxset})\end{bmatrix}\\ \text{cont} & : \begin{bmatrix}\text{x=dgb-params.refset} : \textit{Set}(\textit{Ind})\end{bmatrix}\end{bmatrix}$$

$$\mapsto \text{out:}\begin{bmatrix}\text{cxtype} : \textit{word}\\ \text{phon=F}_{\text{pl}}\text{(in.phon)} : \textit{list}(\textit{Phoneme})\\ \text{cat} : \begin{bmatrix}\text{head} : \begin{bmatrix}\text{pos=n} : \textit{PoS}\\ \text{agr} : \begin{bmatrix}\text{num=pl} : \textit{Num}\end{bmatrix}\\ \text{distr=}- : \textit{Binary}\end{bmatrix}\\ \text{spr} : \left\langle\begin{bmatrix}\text{head} : \begin{bmatrix}\text{count=+} : \textit{Binary}\end{bmatrix}\end{bmatrix}\right\rangle\end{bmatrix}\\ \text{dgb-params} : \begin{bmatrix}\text{refset} & : \textit{Set}(\textit{Ind})\\ \text{maxset} & : \textit{Set}(\textit{Ind})\\ \text{compset} & : \textit{Set}(\textit{Ind})\\ \text{c1} & : \text{union(refset,compset,maxset)}\\ \text{c2} & : \overrightarrow{\textit{PType}}(\text{maxset})\end{bmatrix}\\ \text{cont} : \begin{bmatrix}\text{x=dgb-params.refset} : \textit{Set}(\textit{Ind})\end{bmatrix}\end{bmatrix}$$

Basically, PL-count does three things:

- it replaces the phonetic representation of the input word by its plural form (by means of a phonetic plural function 'F_{pl}');
- it specifies the agreement value of the output to be 'plural' (pl);
- it makes a plural count noun compatible with any plural type (that is, *not* forcing a distributive interpretation by default) *via* feature 'distr=−'.

Due to the lexically specified congruence between a determiner's and a noun's agreement values (see (9)), syntactic agreement precludes singular determiners like *a* or *every* from combining with an output of PL-Count. They can, however, combine with a singular noun, which is the output of SG-count:

(13) **Singular Rule for Count Nouns (SG-count)**

$$\text{in:}\begin{bmatrix}\text{cxtype} & : \textit{lexeme}\\ \text{phon} & : \textit{list}(\textit{Phoneme})\\ \text{cat} & : \begin{bmatrix}\text{head} : \left[\text{pos=n} : \textit{PoS}\right]\\ \text{spr} : \left\langle\left[\text{head} : \left[\text{count=+} : \textit{Binary}\right]\right]\right\rangle\end{bmatrix}\\ \text{dgb-params} & : \begin{bmatrix}\text{refset} & : \textit{Set}(\textit{Ind})\\ \text{maxset} & : \textit{Set}(\textit{Ind})\\ \text{compset} & : \textit{Set}(\textit{Ind})\\ \text{c1} & : \text{union(refset,compset,maxset)}\\ \text{c2} & : \overrightarrow{\textit{PType}}(\text{maxset})\end{bmatrix}\\ \text{cont} & : \left[\text{x=dgb-params.refset} : \textit{Set}(\textit{Ind})\right]\end{bmatrix}$$

$$\mapsto \text{out:}\begin{bmatrix}\text{cxtype} : \textit{word}\\ \text{phon=F}_{\text{sg}}\text{(in.phon)} : \textit{list}(\textit{Phoneme})\\ \text{cat} : \begin{bmatrix}\text{head} : \begin{bmatrix}\text{pos=n} : \textit{PoS}\\ \text{agr} : \left[\text{num=sg} : \textit{Num}\right]\\ \text{distr=}- : \textit{Binary}\end{bmatrix}\\ \text{spr} : \left\langle\left[\text{head} : \left[\text{count=+} : \textit{Binary}\right]\right]\right\rangle\end{bmatrix}\\ \text{dgb-params} : \begin{bmatrix}\text{refset} & : \textit{Set}(\textit{Ind})\\ \text{maxset} & : \textit{Set}(\textit{Ind})\\ \text{compset} & : \textit{Set}(\textit{Ind})\\ \text{c1} & : \text{union(refset,compset,maxset)}\\ \text{c2} & : \overrightarrow{\textit{PType}}(\text{maxset})\\ \text{refind} & : \textit{Ind}\\ \text{c3} & : \text{in(refind,refset)}\end{bmatrix}\\ \text{cont} : \left[\text{x=dgb-params.refind} : \textit{Ind}\right]\end{bmatrix}$$

The main task of SG-count is to introduce a refind into dgb-params and assign it to the noun's content value. As discussed in Lücking and Ginzburg (2022, §3.5), dgb-params are just one set of parameter that accomplish 'referential/quantificational bookkeeping'. There is a set of coercion rules mapping between them, bringing about the different witnessing conditions discussed in Lücking and Ginzburg (2022, §3.5) above. This family of rules is exemplified in (14) by means of the refset. Similar rules obtain for maxset and compset.

(14) **Coercion rules mapping between dgb-params and q-params**

a. in: $\begin{bmatrix} \text{cat} & : \left[\text{pos=np} : \mathit{PoS}\right] \\ \text{dgb-params} & : \mathit{RecType} \\ \text{q-params} & : \left[\text{refset} : \mathit{Set}(\mathit{Ind})\right] \\ \text{cont} & : \left[\text{x=q-params.refset} : \mathit{Set}(\mathit{Ind})\right] \end{bmatrix}$

$\mapsto$ out: $\begin{bmatrix} \text{dgb-params} & : \left[\text{refset} : \mathit{Set}(\mathit{Ind})\right] \\ \text{q-params} & : \mathit{RecType} \\ \text{cont} & : \left[\text{x=dgb-params.refset} : \mathit{Set}(\mathit{Ind})\right] \end{bmatrix}$

b. in: $\begin{bmatrix} \text{cat} & : \left[\text{pos=np} : \mathit{PoS}\right] \\ \text{q-params} & : \mathit{RecType} \\ \text{dgb-params} & : \left[\text{refset} : \mathit{Set}(\mathit{Ind})\right] \\ \text{cont} & : \left[\text{x=dgb-params.refset} : \mathit{Set}(\mathit{Ind})\right] \end{bmatrix}$

$\mapsto$ out: $\begin{bmatrix} \text{q-params} & : \left[\text{refset} : \mathit{Set}(\mathit{Ind})\right] \\ \text{dgb-params} & : \mathit{RecType} \\ \text{cont} & : \left[\text{x=q-params.refset} : \mathit{Set}(\mathit{Ind})\right] \end{bmatrix}$

Not much has to be done for verbs: the corresponding singular and plural rules add the required agreement feature, which, by means of the general verb constraint, has to comply with that of the verb's specifier (i.e., the subject in a sentence).

Both PL-count and SG-count (as well as the corresponding verb rules omitted here for the sake brevity) obey a one-to-one correspondence between syntactic and semantic number. Given evidence such as the following the straightforward syntax-semantics relationship with regard to number seems too simplistic, however:

- There are a couple of plural expressions denoting a single individual such as *pluralis majestatis*, *pluralis modestiae*, *pluralis auctoris* or *pluralis excellentiae*. Though being syntactically plural, their content contribution consists in a semantic object of type *Ind*.
- There are syntactically singular expressions that receive a plural interpretation: *every*-QNPs are a case in point.

We account for the latter case in terms of lexical noun rules which are driven by the *distr* feature (we will not be concerned with the various forms of pluralia mentioned in the first bullet point, however). To this end, a singular count noun with feature *distr*=− and a refind as semantic value can turn into a singular count noun with feature *distr*=+ and a refset as a semantic value:

(15) **Distributive noun rule (dist-n-rule)**

$$\text{in:}\begin{bmatrix}\text{cxtype}: \textit{word} \\ \text{cat}: \begin{bmatrix}\text{head}: \begin{bmatrix}\text{pos=n}: \textit{PoS} \\ \text{agr}: \begin{bmatrix}\text{num=sg}: \textit{Binary}\end{bmatrix} \\ \text{distr=}-: \textit{Binary}\end{bmatrix}\end{bmatrix} \\ \text{dgb-params}: \begin{bmatrix}\text{refset}: \textit{Set}(\textit{Ind}) \\ \text{refind}: \textit{Ind} \\ \text{c3}: \text{in(refind,refset)}\end{bmatrix} \\ \text{cont}: \begin{bmatrix}\text{x=dgb-params.refind}: \textit{Ind}\end{bmatrix}\end{bmatrix}$$

$$\mapsto \text{out:}\begin{bmatrix}\text{cxtype}: \textit{word} \\ \text{cat}: \begin{bmatrix}\text{head}: \begin{bmatrix}\text{pos=n}: \textit{PoS} \\ \text{agr}: \begin{bmatrix}\text{num=sg}: \textit{Binary}\end{bmatrix} \\ \text{distr=}+: \textit{Binary}\end{bmatrix}\end{bmatrix} \\ \text{dgb-params}: \begin{bmatrix}\text{refset}: \textit{Set}(\textit{Ind}) \\ \text{refind}: \textit{Ind} \\ \text{c3}: \text{in(refind,refset)}\end{bmatrix} \\ \text{cont}: \begin{bmatrix}\text{x=dgb-params.refset}: \textit{Set}(\textit{Ind})\end{bmatrix}\end{bmatrix}$$

Of course, the dist-n-rule needs to be echoed by verb phrases: if a verb's specifier exhibits the *distr*=+ feature, the verb receives a distributive interpretation. The latter happens when a *distr*=+ NP merges into a VP's specifier slot when both are to be combined into a sentence by means of the head-subject rule (which is given in the subsequent section 2.4). Dist-v-rule is formulated over phrasal constructions, so that it applies to verbs that are saturated with all but the subject argument.

(16) **Distributive verb rule (dist-v-rule)**

$$\text{in:}\begin{bmatrix}\text{cxtype}: \textit{phrase} \\ \text{cat}: \begin{bmatrix}\text{head}: \begin{bmatrix}\text{pos=v}: \textit{PoS}\end{bmatrix} \\ \text{spr}: \left\langle \begin{bmatrix}\text{head}: \begin{bmatrix}\text{pos=n}: \textit{PoS} \\ \text{distr=}+: \textit{Binary}\end{bmatrix} \\ \text{spr}: \langle\ \rangle \\ \text{cont}: \begin{bmatrix}\text{x}: \textit{Set}(\text{Ind}) \\ \text{y}: \textit{Set}(\textit{Ind})\end{bmatrix}\end{bmatrix} \right\rangle\end{bmatrix}\end{bmatrix}$$

$$\mapsto \text{out:} \begin{bmatrix} \text{cxtype} : \textit{phrase} & \\ \text{cat} : & \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=v} : \textit{PoS} \end{bmatrix} \\ \text{spr} : \left\langle \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=n} : \textit{PoS} \\ \text{distr=+} : \textit{Binary} \end{bmatrix} \\ \text{spr} : \langle\ \rangle \\ \text{cont} : \begin{bmatrix} \text{x} : \textit{Set}(\text{Ind}) \\ \text{y} : \textit{Set}(\textit{Ind}) \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{cont} : & \begin{bmatrix} \text{nucl} : \overrightarrow{\textit{PType}}^{\text{dist}}(/\text{cat.spr.cont.x}) \\ \text{anti-nucl} : \neg\overrightarrow{\textit{PType}}^{\text{dist}}(/\text{cat.spr.cont.y}) \end{bmatrix} \end{bmatrix}$$

2.4 Syntactic rules

The syntactic rules that govern the construction of phrasal types from words crucially involve a division of tasks:

- unification (resp. merge types) and tag types account for lexically specified information like syntactic agreement (spr-verb, det-n, ...);
- functional application of cont values in sentential types implements a predicational semantics of subject-verb constructions, as motivated in Lücking and Ginzburg (2022, §2.4, §4.5).

The construction of NPs out of a (quantificational) determiner and a noun is licensed by the *head-determiner rule* (hd-det-rule). Since it is a headed rule, it is merged with the general constraints from headed structures given in (8) above.

The hd-det-rule from Fig. 1 can be instantiated by the QNP *every student*, but not without further ado: since *every* selects for a head noun which is "*distr*=+" but singular *student* has "*distr*=−" due to SG-count, *student* has to undergo the dist-n-rule first, which introduces the appropriate *distr* value and assigns the refset to the content value. After this lexical operation, the hd-det-rule applies to a QNP such as *every student* as shown in Fig. 2.

Resolving the path equalities in (16), the grammatical model of the QNP *every student* (i.e., the mother node from (16)) is as follows:

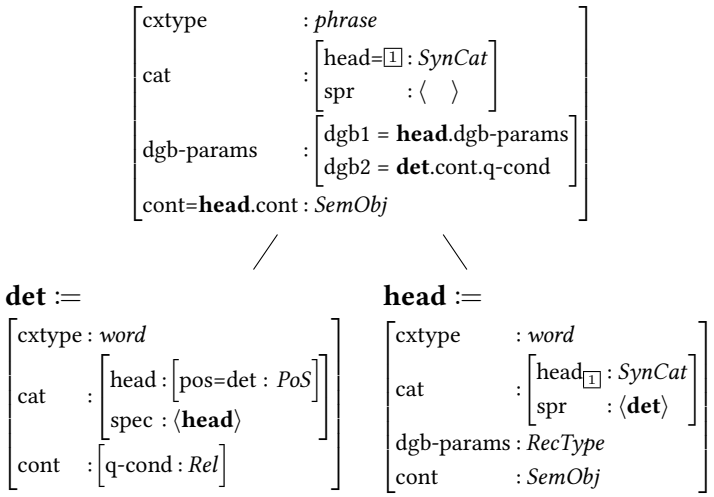


Figure 1: Head-Determiner Rule (hd-det-rule)

(17)
$$\begin{bmatrix}
\text{cxtype} & : \textit{phrase} \\
\text{phon} & : \textit{List}(\text{every, student}) \\
\text{cat} & : \begin{bmatrix} \text{head} : \begin{bmatrix} \text{pos=n} : \textit{PoS} \\ \text{agr} : \begin{bmatrix} \text{pers=3rd} & : \textit{Per} \\ \text{num=sg} & : \textit{Num} \\ \text{gen=masc} & : \textit{Gen} \end{bmatrix} \\ \text{distr=+} : \textit{Binary} \end{bmatrix} \\ \text{spr} : \langle \ \rangle \end{bmatrix} \\
\text{dgb-params} & : \begin{bmatrix} \text{maxset} & : \textit{Set}(\textit{Ind}) \\ \text{refset} & : \textit{Set}(\textit{Ind}) \\ \text{compset} & : \textit{Set}(\textit{Ind}) \\ \text{c1} & : \text{union(refset,compset,maxset)} \\ \text{c2} & : \overrightarrow{\text{student}}(\text{maxset}) \\ \text{refind} & : \textit{Ind} \\ \text{c3} & : \text{in(refind,refset)} \\ \text{q-cond} & : |\text{refset}| = |\text{maxset}| \end{bmatrix} \\
\text{cont} & : \begin{bmatrix} \text{x = dgb-params.refset} \\ \text{y = dgb-params.compset} \end{bmatrix}
\end{bmatrix}$$

Now, if the NP from (17) merges into the spr field of a verb phrase by instantiating the singular head-subject-rule, a semantic error will occur, since a singular predicate expects a subject's content of type *Ind* but *every student* provides one (in fact, two) of type *Set*(*Ind*). A little detour will result in a successful parse, namely when the verb phrase passes through dist-v-rule first. Now *every student* und the intransitive verb *runs* can enter the plural head-subj-rule. Note that both the subject NP as well as the VP are syntactically singular but nonetheless instantiate a plural sentential rule. In other words, the head-sbj-rule is a *semantic*, not a *syntactic* rule

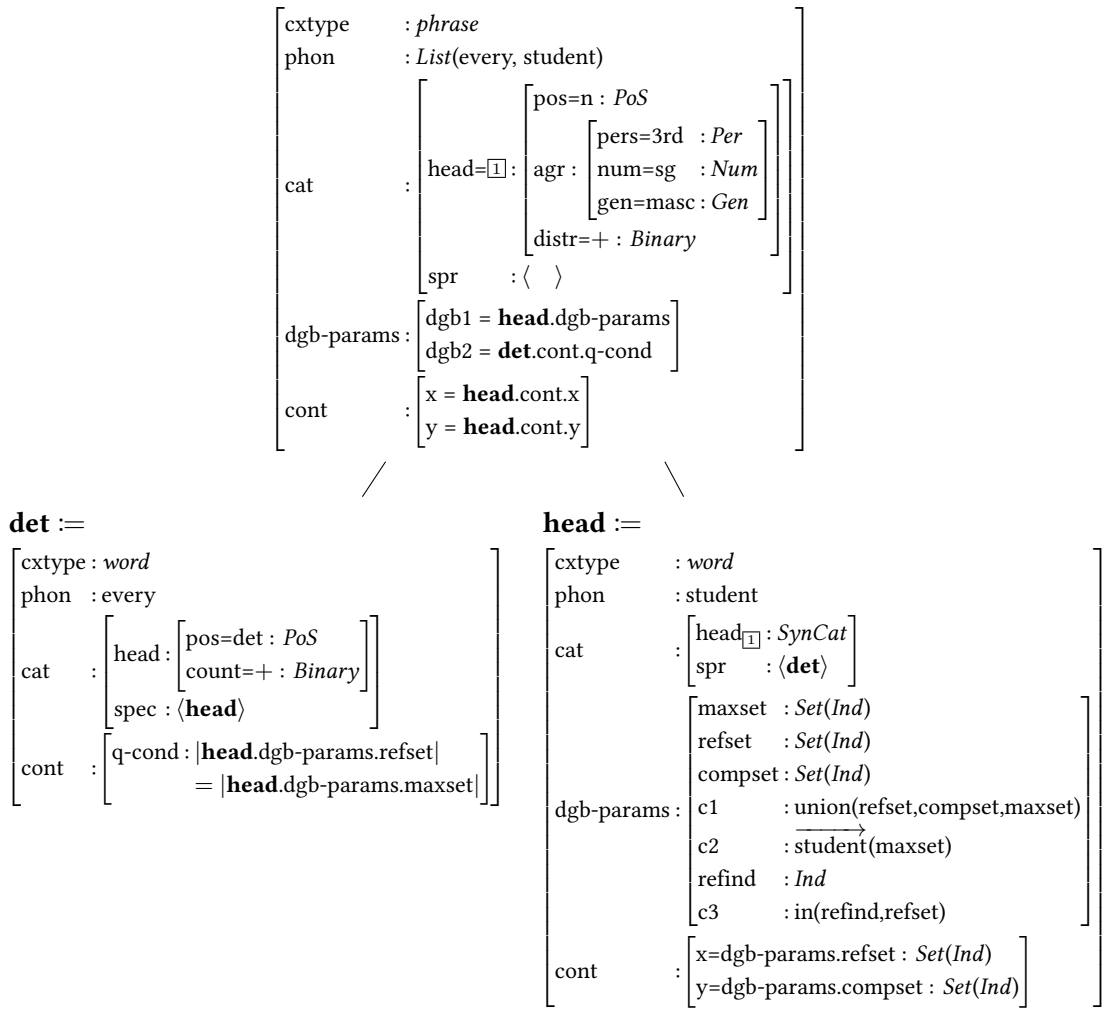


Figure 2: Instantiating the hd-det-rule with the QNP *every student*.

(the VP's content is of type $\overrightarrow{\text{IV}}$, which is the plural type of an intransitive verb). The plural head-subject rule (pl-hd-subj-rule) is shown in Fig. 3.

Combining *every student* (as **subj**) and *runs* (as **head**) by dint of the plural head-subject-rule gives rise to a mother structure from Fig. 4 (where path equalities, for example, between subject's cont and dgb-params, are resolved). The type in Fig. 4 represents a fully referential interpretation, since all reference markers are grounded within dgb-params. Moving elements from dgb-params to q-params is lexically licensed by rules that operate on NPs (see the sample coercion rules given in section 2.3). These rules form a family whose members corresponds to the pragmatic processes involved in spreading the reference markers (that is, maxset, refset, compset and refind) over the parameter set appropriate in the context of a particular utterance.

The grammar fragment sketched in this supplement is sufficient for deriving sentences containing both a QNP subject and a QNP object. The final sentence structure of the example sen-

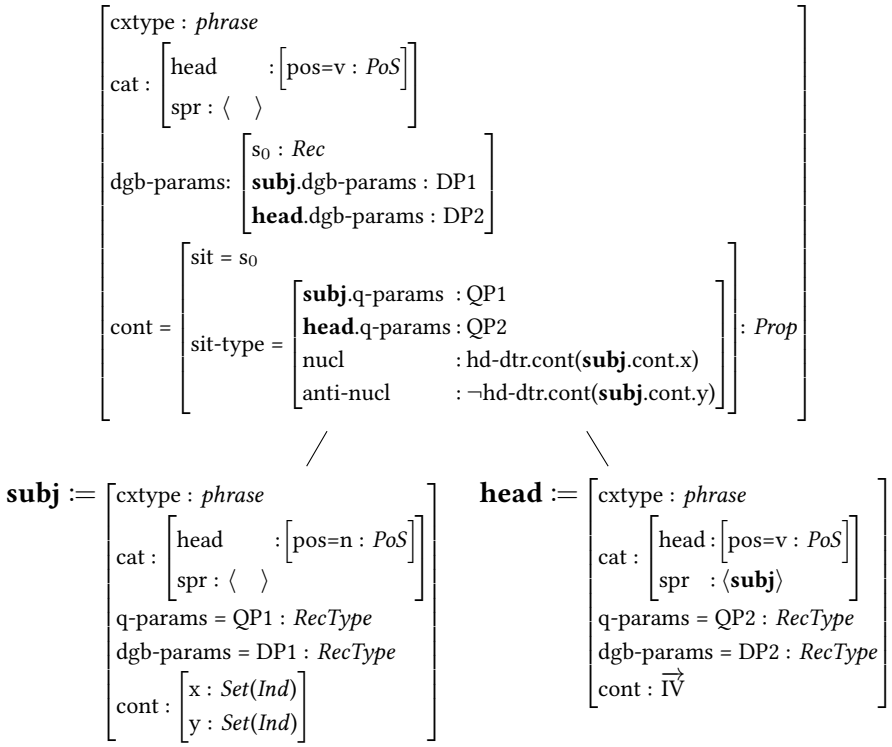


Figure 3: Plural head-subject rule (pl-hd-subj-rule)

tence used there, *Every student lifted the piano*, is given in Fig. 5. Due to *every*, the verb phrase *lifted the piano* represents a distributive plural type according to which the students lifted the piano individually. In accordance with the psycholinguistic evidence reviewed in Lücking and Ginzburg (2022, §1.1), the grammar fragment outlined above dispenses with quantifier floating and implements an *in situ* account. This means that readings attributed to scopal relations between QNPs within a sentence have to be replaced by other, more psycholinguistic and processing oriented mechanisms. In this regard we simply note finally that structures like that in (17) captures so-called wide-scope readings—which turn out to be relational meanings. So-called narrow-scope readings are modelled in terms of dependent functions, cf. Lücking and Ginzburg (2022, §5).

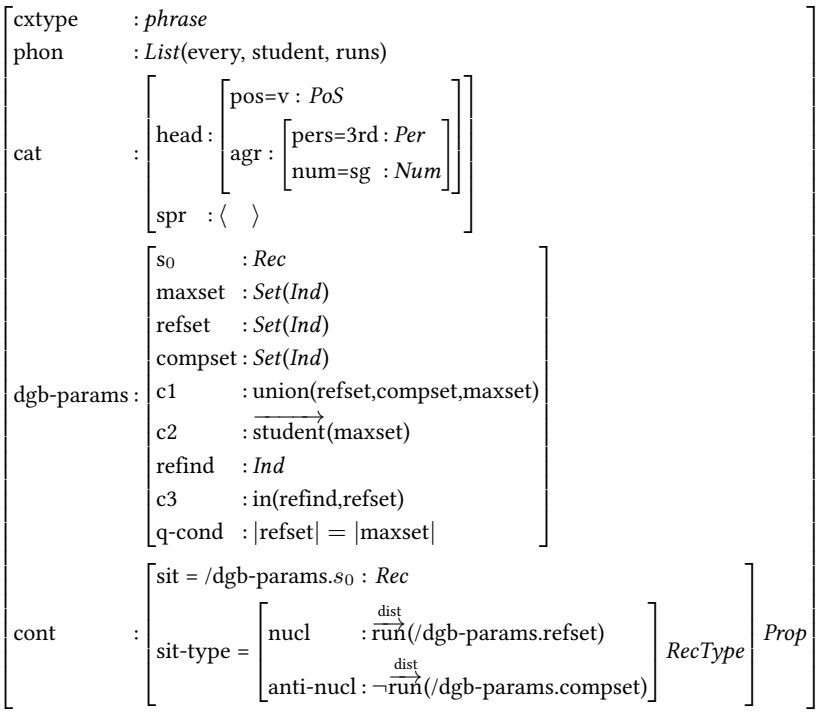


Figure 4: Applying the pl-hd-subj-rule to *Every student runs.*

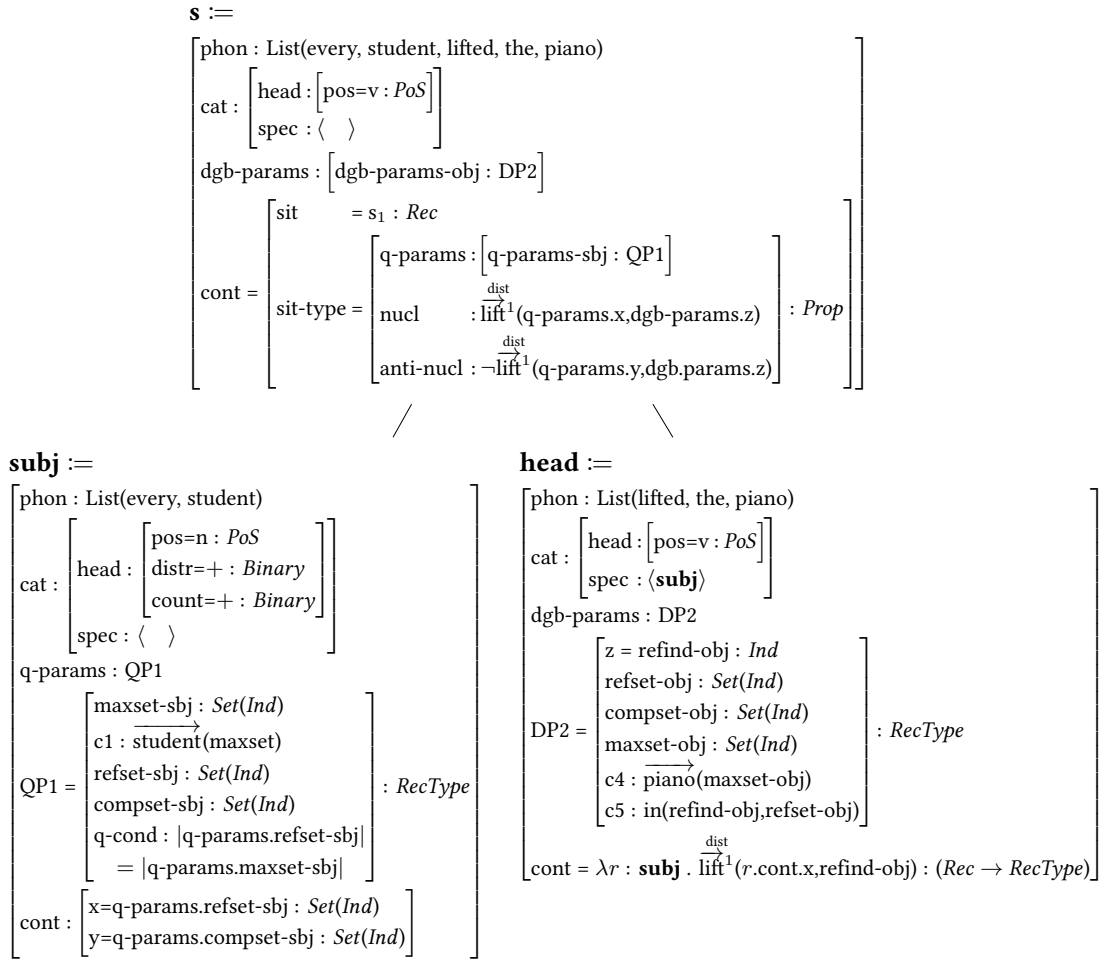


Figure 5: Deriving the transitive sentence *Every student lifted the piano.*