Constraining MGbank: Agreement, L-Selection and Supertagging in Minimalist Grammars

John Torr
School of Informatics
University of Edinburgh
11 Crichton Street, Edinburgh, UK
john.torr@cantab.net

Abstract

This paper reports on two strategies that have been implemented for improving the efficiency and precision of wide-coverage Minimalist Grammar (MG) parsing. The first extends the formalism presented in Torr and Stabler (2016) with a mechanism for enforcing fine-grained selectional restrictions and agreements. The second is a method for factoring computationally costly null heads out from bottom-up MG parsing; this has the additional benefit of rendering the formalism fully compatible for the first time with highly efficient Markovian supertaggers. These techniques aided in the task of generating MGbank, the first wide-coverage corpus of Minimalist Grammar derivation trees.

1 Introduction

Parsers based on deep grammatical formalisms, such as CCG (Steedman and Baldridge, 2011) and HPSG (Pollard and Sag, 1994), exhibit superior performance on certain semantically crucial (unbounded) dependency types when compared to those with relatively shallow context free grammars (in the spirit of Collins (1997) and Charniak (2000)) or, in the case of modern dependency parsers (McDonald and Pereira (2006), Nivre et al. (2006)), no explicit formal grammar at all (Rimell et al. (2009), Nivre et al. (2010)). As parsing technology advances, the importance of correctly analysing these more complex construction types will also inevitably increase, making research into deep parsing technology an important goal within NLP.

One deep grammatical framework that has not so far been applied to NLP tasks is the Minimalist Grammar (MG) formalism (Stabler, 1997). Linguistically, MG is a computationally-oriented formalization of many aspects of Chomsky's (1995) Minimalist Program, arguably still the dominant framework in theoretical syntax, but so far conspicuously absent from NLP conferences. Part of the reason for this has been that until now no Minimalist treebank existed on which to train efficient statistical Minimalist parsers.

The Autobank (Torr, 2017) system was designed to address this issue. It provides a GUI for creating a wide-coverage MG together with a module for automatically generating MG trees for the sentences of the Wall Street Journal section of the Penn Treebank (PTB) (Marcus et al., 1993), which it does using an exhaustive bottom-up MG chart parser[1]. This system has been used to create MGbank, the first wide coverage (precision-oriented) Minimalist Grammar and MG treebank of English, which consists of 1078 hand-crafted MG lexical categories (355 of which are phonetically null) and currently covers approximately half of the WSJ PTB sentences. A problem which arose during its construction was that without any statistical model to constrain the derivation, MG parsing had to be exhaustive, and this presented some significant efficiency challenges once the grammar grew beyond a certain size[2], mainly because of the problem of identifying the location and category of phonetically silent heads (equivalent to type-changing unary rules) allowed by the theory. This problem was particularly acute for the MGbank grammar, which makes extensive use of such heads to multiply out the lexicon during pars-

[1] The parser is based on Harkema's (2001) CKY variant.

[2] As Cramer and Zhang (2010) (who pursue a similar treebanking strategy for HPSG) observe, there is very often considerable tension between the competing goals of efficiency and coverage for deep, hand-written and precision-oriented parsers, which aim not only to provide detailed linguistic analyses for grammatical sentences, but also to reject ungrammatical ones wherever possible.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 590–600
Melbourne, Australia, July 15 - 20, 2018.

ing. This approach reduces the amount of time needed for manual annotation, and also enables the parser to better generalise to unseen constructions, but it can quickly lead to an explosion in the search space if left unconstrained.

This paper provides details on two strategies that were developed for constraining the hypothesis space for wide-coverage MG parsing. The first of these is an implementation of the sorts of selectional restrictions[3] standardly used by other formalisms, which allow a head to specify certain fine-grained properties about its arguments. Pesetsky (1991) refers to this type of fine-grained selection as *l(exical)-selection*, in contrast to coarser-grained *c(ategory)-selection* and semantic *s-selection*. The same system is also used here to enforce morphosyntactic agreements, such as subject-verb agreement[4] and case 'assignment'. It is simpler and flatter than the structured feature value matrices one finds in formalisms such as HPSG and LFG, which arguably makes it less linguistically plausible. However, it is also considerably easier to read and to annotate, which greatly facilitated the manual treebanking task.

The second technique to be presented is a method for extracting a set of complex overt categories from a corpus of MG derivation trees which has the dual effect of factoring computationally costly null heads out from parsing (but not from the resulting parse trees) and rendering MGs fully compatible for the first time with existing supertagging techniques. Supertagging was originally introduced in Bangalore and Joshi (1999) for the Lexicalised Tree Adjoining Grammar (LTAG) formalism (Schabes et al., 1988), and involves applying Markovian part-of-speech tagging techniques to strongly lexicalised tag sets that are much larger and richer than the 45 tags used by the PTB. Because each supertag contains a great deal of information about the syntactic environment of the word it labels, such as its subcategorization frame, supertagging is sometimes referred to as 'almost parsing'. It has proven highly effective at making CCG (Clark and Curran, 2007; Lewis et al., 2016; Xu, 2016; Wu et al., 2017) parsing in particular efficient enough to support large-scale NLP tasks, making it desirable to apply this technique to MGs. However, existing supertaggers can only tag what they can see, presenting a problem for MGs, which include phonetically unpronounced heads. Our extraction algorithm addresses this by anchoring null heads to overt ones within complex LTAG-like supertag categories.

The paper is arranged as follows: section 2 gives an informal overview of MGs; section 3 introduces the selectional mechanisms and shows how these are used in MGbank to enforce case 'assignment' (3.1), l-selection (3.2) and subject-verb agreement (3.3); section 4 presents the algorithm for extracting supertags from a corpus of MG derivation trees (4.1), gives details of how a standard CKY MG parser can straightforwardly be adapted to make use of these complex tags (4.2), and presents some preliminary supertagging results (4.3) and a discussion of these (4.4); section 5 concludes the paper.

2 Minimalist Grammars

For a more detailed and formal account of the MG formalism assumed in this paper, see Torr and Stabler (2016) (henceforth T&S); here we give only an informal overview. MG is introduced in Stabler (1997); it is a strongly lexicalised formalism in which categories are comprised of lists of structure building features ordered from left to right. These features must be checked against each other and deleted during the derivation, except for a single c feature on the complementizer (C) heading the sentence, which survives intact (equivalent to reaching the S root in classical CFG parsing). Features are checked and deleted via the application of a small set of abstract Merge and Move rules. Two simple MG lexical entries are given below (The :: is a type identifier[5]):

him :: d
helps :: d= v

The structure building features themselves can be categorized into four classes: selector =x/x= features, selectee x features, licensor +y features, and licensee -y features. In a directional MG, such as that presented in T&S, the = symbol on the selector can appear on either side of the x category symbol, and this indicates whether selection is to the left or to the right. For instance, in our toy lexicon *helps*'s first feature is a d= selector, indicating that it is looking for a DP on its right. Since the

[3] These were briefly introduced in Torr (2017), but are expounded here in much greater depth.

[4] The approach to agreement adopted here differs in various respects from the operation Agree (Chomsky (2000) (2001)) assumed in current mainstream Minimalism.

[5] :: indicates a non-derived item and : a derived one.

first feature of *him* is a d selectee, we can merge these two words to obtain the following VP category, where ϵ is the empty string (The reason for the commas separating the left and right dependent string components from the head string component is to allow for subsequent head movement of the latter (see Stabler (2001)):

ϵ, *helps, him* : v

The strings of the two merged elements have been here been concatenated, but this will not always be the case. In particular, if the selected item has additional features behind its selectee, then it will need to check these in subsequent derivational steps via applications of Move. In that case the two constituents must be kept separate within a single expression following Merge. To illustrate this, we will update the lexicon as follows:

him :: d -case
helps :: d= +CASE v

Merging these two items results in the following expression:

ϵ, *helps*, ϵ : +CASE v, *him* : -case

The two subconstituents, separated above by the rightmost comma, are referred to as *chains*; the leftmost chain in any expression is the head of the expression; all other chains are movers. The +CASE licensor on the head chain must now attract a chain within the expression with a matching -case licensee as its first feature to move overtly to its left dependent (specifier) position[6]. Exactly one moving chain *must* satisfy this condition, or this expression will be unable to enter into any further operations (if more than one chain has the same licensee feature, it will violate a constraint on MG derivations known as the *Shortest Move Constraint* (SMC) and automatically be discarded). As this condition is satisfied by just *him*'s -case feature, we can perform the unary operation Move on this expression, resulting in the following new, single-chained expression:

him, helps, ϵ : v

We can represent these binary Merge and unary Move operations using the MG derivation tree in fig 1a. Derivation trees such as this are used frequently in work on Stablerian Minimalist Grammars, but they can be deterministically mapped into phrase structure trees like fig 1b[7].

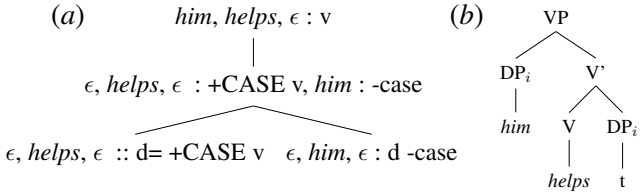

Figure 1: An MG Derivation tree for the VP *him, helps* (a); and its corresponding Xbar phrase structure tree (b). At this stage in the derivation the verb and its object are incorrectly ordered. This will be rectified by subsequent V-to-v head movement placing the verb to the left of its object.

To continue this derivation and derive the transitive sentence *he helps him*, we will expand our lexicon with the following categories, where square brackets indicate a null head and a > diacritic on a selector feature indicates that a variant of Merge is triggered in which the head string of the selected constituent undergoes head movement to the left of the selecting constituent's head string:

he :: d -case
[trans] :: >v= =d lv[8]
[pres] :: lv= +CASE t
[decl] :: t= c

The full derivation tree and corresponding Xbar phrase structure tree for the sentence are given in fig 2 and fig 3 respectively.

3 Case, L-selection and Agreement

3.1 Case 'Assignment'

Notice that at present both the nominative and accusative forms of the masculine personal pronoun

[6]Uppercase licensors specify overt movement; lowercase licensors, by contrast, trigger covert movement, where only the features move, not the string (see T&S). Note that the MGbank grammar follows Chomsky's (2008) suggestion that it is the lexical verb V, rather than the null 'little v' head governing it, which checks the object's features, having inherited the relevant licensors (offline we assume) from v. This unifies the analysis of standard transitives with ECM constructions (*Jack expected Mary to help*), which in MGbank involve overt raising of the subject of the embedded infinitival clause to spec-VP to check accusative case (object control *Jack persuaded Mary to help* involves two such movements, the first for theta and the second for case).

[7]MGbank includes MG derivation tree, MG derived (bare phrase structure) tree, and Xbar tree formats.

[8]Note that little v is written as lv in MGbank derivation trees because upper vs lowercase letters are used to trigger different rules. In the corresponding MGbank Xbar trees, however, v has been converted to V and lv to v, to make these trees more familiar.

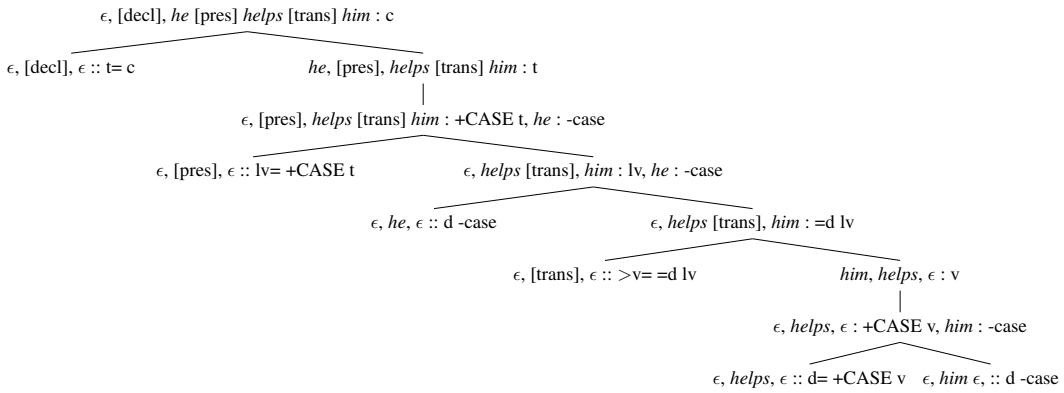

Figure 2: MG derivation tree for the sentence *he helps him*.

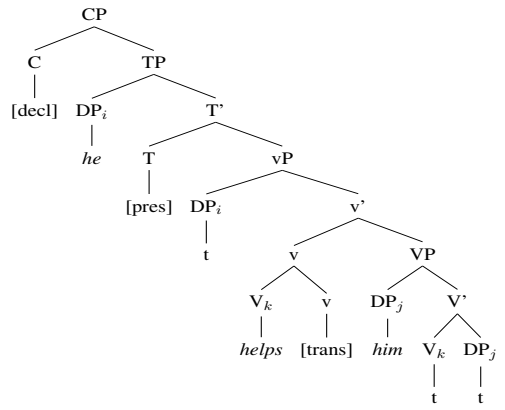

Figure 3: Xbar phrase structure tree for the sentence *he helps him*.

in our lexicon have the same feature sequence. This means that as well as correctly generating *he helps him*, our grammar also overgenerates *him helps he*. One way to solve this would be to split +/-case features into +/-nom and +/-acc. However, many items of category d in English (e.g. *the*, *a*, *you*, *there*, *it*) are syncretised (i.e. have the same phonetic form) for nominative vs. accusative case. This solution therefore lacks elegance as it expands the lexicon with duplicate homophonic entries differing in just a single (semantically meaningless) feature. Furthermore, increasing the size of the set k of licensees could adversely impact parsing efficiency, given that the worst case theoretical time complexity of MG chart parsing is known to be n^{2k+3} (Fowlie and Koller, 2017), where k is the number of moving chains allowed in any single expression by the grammar.

Instead, we will retain the single -case licensee feature and introduce NOM and ACC as subcategories, or *selectional properties*, of this feature. We will also subcategorize licensor features using *selectional requirements* of the form +X and -X, where X is some selectional property. Positive +X features require the presence of the specified property on the licensee feature being checked, while -X features require its absence. For example, consider the following updated lexical entries, where individual selectional features are separated by the . symbol:

him :: d -case{ACC}

he :: d -case{NOM}

helps :: d= +CASE{+ACC} v{PRES.TRANS}

[pres] :: lv{+PRES}= +CASE{+NOM} t{FIN.PRES}

[trans] :: >v{+TRANS}= =d lv

The +ACC selectional requirement on the V head's +CASE licensor specifies that the object's licensee feature must bear an ACC selectional property, while +NOM on the T(ense) head indicates that the subject's licensee must have a NOM property. For SMC purposes, however, these two different subcategories of -case will still block one another, meaning that k remains unaffected. The reader should satisfy themselves that our grammar now correctly blocks the ungrammatical *him helps he*.

We can now also address the aforementioned syncretism issue without increasing the size of the grammar. To do this, we simply allow features to bear multiple selectional properties from the same paradigm. For example, representing the pronoun *it* as follows will allow it to appear in either a nominative or an accusative case licensing position:

it :: d -case{ACC.NOM}

3.2 L-selection

As well as constraining Move, selectional restrictions can also constrain Merge. For instance, we can ensure that a subject control verb like *want* subcategorizes for a *to*-infinitival CP complement, and thereby avoid overgenerating *Jack wants that she help(s)*, simply by using the following categories for *want* and *that*:

want :: c{+INF}= v{TRANS}
that :: t{+FIN}= c{DECL.FIN}

Because *that* lacks the INF feature required by *want*, the ungrammatical derivation is blocked. We also need to block **Jack wants she help(s)*, where the overt C head is omitted. Minimalists assume that finite embedded declaratives lacking an overt C are nevertheless headed by a null C - a silent counterpart of *that*. A complicating factor is that a null complementizer is also assumed to head certain types of embedded infinitivals, including the embedded *help* clause in *Jack wants* [$_{CP}$ *to help*]. Given that these null C heads are (trivially) homophones and that they arguably exist to encode the same illocutionary force[9], an elegant approach would be to minimize the size of the lexicon - and hence the grammar - by treating them as one and the same item. On the other hand, using a single null C head syncretised with both FIN and INF will fail to block **Jack wants she help(s)*.

At present both C and T are specified as FIN, suggesting a redundancy. Instead, therefore, we will assume that T, being the locus of tense, is also the sole locus of inherent finiteness, but that C's selectee may inherit FIN or INF from its TP complement as the derivation proceeds[10]. Only a null C which inherits INF from a *to*-TP complement will be selectable by a verb like *want*, blocking the ungrammatical **Jack wants she help(s)*. However, although lacking inherent tense *properties*, certain C heads continue to bear inherent tense *requirements*[11]; for instance, *that*'s selector will retain its inherent +FIN, identifying it as a *finite* complementizer.

To implement this percolation[12] mechanism, we now introduce *selectional variables*, which we write as x, y, z etc. A variable on a selector or licensor feature will cause all the selectional properties and requirements (but not other variables) contained on the selectee or licensee feature that it checks to be copied onto all other instances of that variable on the selecting or licensing category's remaining unchecked feature sequence. Consider the following:

[trans] :: >v{+TRANS.x}= =d lv{x}
[pres] :: lv{+PRES.x}= +CASE{+NOM.x} t{FIN.x}
to :: lv{+BARE.x}= t{INF.x}
[decl] :: t{x}= c{DECL.x}
that :: t{+FIN.x}= c{DECL.x}

The [pres] T head has an x variable on its lv= selector feature and this same variable also appears to the right on its +CASE licensor and t selectee; any selectional properties or requirements contained on the lv selectee of its vP complement will thus percolate onto these two features (see fig 4). The x's on the two C heads will percolate the FIN property from the t selectee of [pres] to the c selectee of [decl], where it can be selected for by a verb like *say*, but not *want*, which requires INF (contained on the *to* T head); this will correctly block **Jack wants (that) she help(s)*.

Although we will not discuss the details here, it is worth noting that the MGbank grammar also uses this same percolation mechanism to capture long distance subcategorization in English subjunctives, thereby allowing *Jack demanded that she be there on time* while also blocking **Jack demanded that she is there on time*.

[9] Infinitival complementizers are sometimes assumed to encode *irrealis* force (see e.g. Radford (2004)) in contrast to *that* and its null counterpart which encode declarative force. However, the fact that *Jack expects her to help* is (on one reading) virtually synonymous with *Jack expects that she will help* suggests that in both cases the C head is encoding the same semantic property, with any subtle difference in meaning attributable to the contents of the Tense (T) head (i.e. *to* vs. *will*). Consider also *Mary wondered whether to help* vs. *Mary wondered whether she should help*, where the embedded infinitival and finite clauses are both clearly interrogative.

[10] If Grimshaw (1991) is correct that functional projections like DP, TP and CP are part of *extended projections* of the N and V heads they most closely c-command, then we should not be surprised to find instances where fine-grained syntactic properties are projected up through these functional layers.

[11] The property vs. requirement distinction mirrors Chomsky's (1995) interpretable vs. uninterpretable one.

[12] Note that because we are only allowing selectional properties and requirements to percolate, rather than the structure building feature themselves, this system is fundamentally different from that described in Kobele (2005), where it was shown that allowing licensee features to be percolated leads to type 0 MGs. Furthermore, by unifying any multiple instances of the same selectional property or requirement that arise on a structure building feature owing to percolation, we can ensure that the set of MG terminals and non-terminals remains finite and thus that the weak equivalence to MCFG (Michaelis, 1998; Harkema, 2001) is maintained.

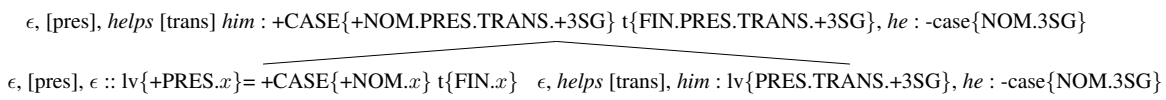

Figure 4: Merge of T with vP with percolation of selectional properties and requirements.

3.3 Subject-Verb Agreement

The percolation mechanism introduced above can also be used to capture agreement between the subject and the inflected verb. In Minimalist theory, this agreement is only indirect: the subject actually agrees directly with T when it moves to become the latter's specifier, having been initially selected for either by V (in the case of non-agent arguments) or by v (in the case of agent subjects - see fig 3)[13]. There is also assumed to be some sort of syntactic agreement (Roberts (2010)) and/or phonetic (Chomsky (2001)) process operating between T and the inflected verb, resulting in any tense/agreement inflectional material generated in T(ense) being suffixed onto the finite verb.

In MGbank, tense agreement is enforced between T and the finite verb by percolating a PRES or PAST selectional property from the selectee of the latter up through the tree so that it can be selected for by the [pres] or [past] T head. Subject-verb agreement, meanwhile, is enforced by also placing an agreement selectional restriction (+3SG, +1PL, -3SG etc) on the finite verb's selectee, and then percolating this up to the +CASE licensor of the T head. We thus have the following updated entries:

him :: d -case{ACC.3SG}
he :: d -case{NOM.3SG}
helps :: d= +CASE{+ACC} v{+3SG.PRES}

The percolation step from little v (lv) to T is shown in fig 4; lv has already inherited PRES and +3SG from V (*helps*) at this point, and these features now percolate to T's licensor and selectee[14] owing to the x variables; the PRES feature inherited from V by v is selected for by T, enforcing non-local tense agreement between T and V, while the +3SG enforces subject verb agreement[15].

4 MG Supertagging

The above selectional system restricts the parser's search space sufficiently well that it is feasible to generate an initial MG treebank for many of the sentences in the PTB, particularly the shorter ones and those longer ones which do not require the full range of null heads to be allowed into the chart[16]. However, for longer sentences requiring null heads such as extraposers, topicalizers or focalizers, parsing remains impractically slow. In this section we show how computationally costly null heads can be factored out from MG parsing al-

[13] A reviewer asks why all subjects are not directly selected for by V, suggesting that this appears to be a deviation from semantics, and more generally calls for some explanation of the underlying modelling decisions adopted here (e.g. head movements, case movements, null heads etc) which clearly deviate from the more surface oriented analyses of other formalisms used in NLP. In many cases these decisions rest on decades of research which we cannot hope to summarise here; for good introductions to Minimalism, see Radford (2004) and Hornstein et al. (2005). It is worth noting, however, that the null v head in fig 3 is essentially a valency increasing causative morpheme which ends up suffixed to the main verb (via head movement of the latter), effectively enabling it to take an additional 'external' argument. We can therefore view the V-v complex as a single synthetic verbal head, so that just as in a language like Turkish the verb *öl* meaning 'to die' can be transformed from an intransitive to a transitive (meaning 'to kill') by appending to it the causative suffix *dür*, in English a verb like *break* can be transformed from an intransitive (*the window broke*) to a transitive (*he broke the window*) by applying a null version of this morpheme. This cross-linguistic perspective (which makes this formalism potentially very relevant for machine translation) reflects a central goal of Minimalism, which is to show that at a relevant level of abstract representation, all languages share a common syntax (making them easier for children to learn). Most of the analyses adopted here are standard ones from the literature (see e.g. Larson's (1988) VP Shell Hypothesis, Baker's (1988) Uniform Theta Assignment Hypothesis, Koopman and Sportiche's (1991) Verb Phrase Internal Subject Hypothesis, and Chomsky (1995; 2008) on little v).

[14] Note that selectional requirements are entirely inert on selectee and licensee features while, conversely, selectional properties are inert on selectors and licensors.

[15] For non-3SG present tense verbs, MGbank uses a -3SG negative selectional requirement; for verbs with more complex paradigms, however, the grammar allows for inclusive disjunctive selectional requirements. For example, the selectee feature of the *was* form of the verb *be* bears the feature [+1SG|+3SG], allowing it to take either a first or third singular subject.

[16] The Autobank parser holds certain costly null heads back from the chart and only introduces these incrementally if it fails to parse the sentence without them. The advantage of this strategy is that it improves efficiency for many sentences, but the disadvantage is that it can also result in correct analyses being bled by incorrect ones. The supertagging approach introduced in this section eliminates this problem, since null heads are now anchored to overt ones as part of complex categories, any of which may freely be assigned by the supertagger.

together by anchoring them to overt heads within complex overt categories extracted from this initial treebank. This allows much more of the disambiguation work to be undertaken by a statistical Markovian supertagger[17], a strategy which has proven highly effective at rendering CCG parsing in particular efficient enough for large-scale NLP tasks. We also show how a standard CKY MG parser can be adapted to make use of these complex categories, and present some preliminary supertagging results.

4.1 Factoring null heads out from MG parsing

Consider again the lexical items which appear along the spine of the clause in fig 2.

[decl] :: t= c
[pres] :: lv= +CASE t
[trans] :: >v= =d lv
helps :: d= +CASE v

Recall that the null [trans] little v merges with the VP headed by overt *helps*, while the null [pres] T head merges with the vP, and the null [decl] C with TP. If we view each of these head-complement merge operations as a link in a chain, then all of these null heads are either directly (in the case of v) or indirectly (in the case of T and C) linked to the overt verb. All of the information represented on V, v, T and C heads in Minimalism is in LTAG represented on a single overt lexical category (known as an initial tree). We can adopt this perspective for Minimalist parsing if we view chains of merges that start with some null head and end with some overt head as constituting complex overt categories. Given a corpus of derivation trees, it is possible to extract all such chains appearing in the corpus, essentially precompiling all of the attested combinations of null heads with their overt anchors into the lexicon. A very simple algorithm for doing this is given below.

for each derivation tree τ:
  for each null head η *in* τ:
    if η *is a proform:*
      linkWithGovernor(η);
    else:
      linkWithHeadOfComplement(η);
  groupLinksIntoSupertags()

[17] During treebank generation we used the C&C (Clark and Curran, 2007) supertagger retrained to take gold CCGbank categories and words as input and output MGbank supertags.

For each derivation tree, we first anchor all null heads either directly or indirectly to some overt head; this is achieved by extracting a set of links, each of which represents one merge operation in the tree. Each link is comprised of the two atomic MG lexical categories that are the arguments to the merge operation along with matching indices indicating which features are checked by the operation. Applying the algorithm to our example sentence would result in the following 3 links:

link1: [decl] :: t=^{1} c, [pres] :: lv= +CASE t^1
link2: [pres] :: lv=^{2} +CASE t, [trans] :: v= =d lv^2
link3: [trans] :: v=^{3} =d lv, *helps* :: d= +CASE v^3

The majority of null heads are simply linked with the head of their complement, the only exception being that null proforms, such as PRO in arbitrary control constructions[18] (named [pro-d] in MGbank) and the null verbal heads used for VP ellipsis ([pro-v] in MGbank), are linked to whichever head selects for them (i.e. their governor). Assuming that null proforms are the only null heads appearing at the bottom of any extended projection (ep)[19] in the corpus, this ensures that all of the lexical items inside a given supertag are part of the same ep, except for PRO, which is trivially an ep in its own right and must therefore be anchored to the verb that selects it. Note that some atomic overt heads (such as *he* and *him* in our example sentence) will not be involved in any links and will therefore form simplex supertags.

Once the merge links and unattached overt heads are extracted, the algorithm then groups them together in such a way that any lexical items which are chained together either directly or indirectly by merge links are contained in the same group. Because links are only formed between null heads and their complements (except in the case of the null proform heads), and not between heads and specifiers or adjuncts, each chain ends with the first overt head encountered, so that every (null or overt) head is guaranteed to appear in just one group and each group is guaranteed to contain at most one overt lexical item.

The above merge links would form one group, or supertag, represented compactly as follows:

[18] Other instances of control are treated as cases of A-movement following Boeckx et al. (2010).

[19] Here, we define the clausal extended projection as running from V up to the closest CP (or TP if CP is absent, as in ECM constructions), and for nominals from N up to the closest PP (or DP if PP is absent).

[decl] :: t=^{1} c
[pres] :: lv=^{2} +CASE t^1
[trans] :: v=^{3} =d lv^2
helps :: d= +CASE v^3

All of the subcategorization information of the main verb is contained within this supertag, but unlike in the case of LTAG categories, this is not always the case: if an auxiliary verb were present between little vP and TP, for instance, then only little v would be anchored to the main verb, while T and C would be anchored to the structurally higher auxiliary. C is the head triggering A'-movements, such as wh-movement and topicalization. A consequence of this is that, although like LTAG (but unlike CCG) A'-movement is lexicalised onto an overt category here, that overt category is often structurally and linearly much closer to the A'-moved element than in LTAG. For instance, in the sentence *what did she say that Pete eats for breakfast?*, an LTAG would precompile the wh-movement onto the supertag for *eats*, whereas here the [int] C head licensing this movement would be precompiled onto *did*.

As noted in Kasai et al. (2017), LTAG's lexicalisation of unbounded A'-movement is one reason why supertagging has proven more difficult to apply successfully to TAG than to CCG, Markovian supertaggers being inherently better at identifying local dependencies. We hope that lexicalising A'-movement into a supertag that is linearly closer to the moved item will therefore ultimately prove advantageous.

4.2 Adapting an existing CKY MG parser to use MG supertags

The MG supertags can be integrated into an existing CKY MG parser quite straight forwardly as follows: first, for each supertag token assigned to each word in the sentence, we map the indices that indicate which features check each other into globally unique identifiers. This is necessary to ensure that different supertags and different instances of the same supertag assigned to different words are differentiated by the system. Then, whenever one of the constrained features is encountered, the parser ensures that it is only checked against the feature with the matching identifier. The parser otherwise operates as usual except that thousands of potential merge operations are now disallowed, with the result that the search space is drastically reduced (though this of course depends on the number of supertags assigned to each word).

One complication concerns the dynamic programming of the chart. In standard CKY MG parsing, as with classical CFG CKY, items with the same category spanning the same substring are combined into a single chart entry during parsing. This prevents the system having to create identical tree fragments multiple times. But the current approach complicates this because many items now have different predetermined futures (i.e. their unchecked features are differentially constrained), and when the system later attempts to reconstruct the trees by following the backpointers, things can become very complicated. We can avoid this issue, however, simply by treating the unique identifiers that were assigned to certain selector features as part of the category. This has the effect of splitting the categories and will, for instance, prevent two single chain categories =d^1 d= v and =d^2 d= v from being treated as a single chart entry until their =d features have been checked.

4.3 Preliminary Results

An LSTM supertagger similar to that in (Lewis et al., 2016) was trained on 13,000 sentences randomly chosen from MGbank, extracting various types of (super)tag from the derivation trees. A further 742 sentences were used for development, and 753 for testing, again randomly chosen. We tried training on just the automatically generated corpus and testing on the hand-crafted trees, but this hurt 1-best performances by 2-4%, no doubt owing to the fact that this hand-crafted set deliberately contains many of the rarer constructions in the Zipfian tail which didn't make it into the automatically generated corpus[20]. With more data this effect should lessen. The results for n-best supertagging accuracies are given in table 1.

4.4 Discussion

Unsurprisingly, the accuracies improve as the number of tags decreases. The CCGbank data contains by far the least tag types and has the highest performance. However, it is worth noting that the MG supertags contain a lot more information than their CCGbank counterparts, even once A'-movement and selectional restrictions are removed. For example, MGbank encodes all predicate-argument relations directly in the syntax, distinguishing for instance between subject

[20]There are 831 category types in the automatically generated corpus from a total of 1078 for the entire treebank.

	rei	ab	rei-A'	ab-A'	ov	ccg
|tags|	3087	2087	1883	1181	717	342
1-best	79.1	81.1	83.0	84.2	88.0	92.4
2-best	88.4	90.2	91.1	91.9	95.3	97.1
3-best	91.6	93.5	94.1	94.8	97.1	98.3
10-best	96.4	97.4	97.9	98.2	99.2	99.5
25-best	97.6	98.5	98.9	99.1	99.7	99.7
40-best	98.0	98.7	99.0	99.4	99.8	99.8

Table 1: Accuracies on different MG (super)tag types showing the % of cases where the correct tag appears in the n-best list. The first row gives the number of different (super)tag types in the data; rei(fied) is supertags with all selectional properties and requirements; ab(stract) is supertags with all but 5 of these features removed[22]; -A' indicates that null C heads, and [focalizer], [topicalizer], [wh] and [relativizer] heads were not included in the supertags, thereby delexicalising A'-movement and moving the formalism towards CCG; ov(ert) is the (reified) atomic overt tags; ccg is the ccgbank supertags.

raising and subject control verbs, and between object raising (ECM) and object control verbs, whereas CCGbank itself does not. For a fairer comparison, therefore, we would need to combine CCGbank syntactic types with the semantic types of Bos (Bos et al., 2004). There are also many types of dependencies, such as those for rightward movement and correlative focus (*either..or, neither..nor, both..and*), which could be delixicalised to reduce the size of the supertag sets further. Of course, the more null heads that are allowed freely into the chart, the stronger the statistical model of the derivation itself must be. Finally, the MGbank grammar (particularly in its reified versions) is precision-oriented, in the sense that it blocks many ungrammatical sentence types (agreement/l-selection violations, binding theory violations, (anti)*that*-trace violations, wh-island violations etc). The extra information needed to attain this precision expands the tag set but should also ultimately help in pruning the search space, enabling the parser to try more tags. The CCGbank grammar, meanwhile, is much more flexible (making it very robust), and therefore leaves a much greater proportion of the task of constraining the search space to the probability model.

The 1-best accuracies are clearly not high enough to be practical for wide-coverage MG parsing at present. By the time the 3-best supertags per word are considered, however, the accuracies are in all cases quite high, and by the 25-best they are very high, although it is difficult to say at this point what level will be sufficient for wide-coverage parsing. The overt atomic tagging is much better, achieving high accuracy by the 3-best, but these tags contain the least information and therefore leave much more disambiguation to the parsing model. Clearly, using MG supertags will require an algorithm that navigates the search space as efficiently as possible and allows the supertagger to try as many tags for each word as possible. We are in the process of re-implementing the A* search algorithm of (Lewis and Steedman, 2014), which allows their CCG parser to consider the complete distribution of 425 supertags for each word.

The potential efficiency advantages of parsing with MG supertags are considerable: reparsing the seed set of 960 trees (which includes 207 sentences which were added to cover some constructions not found in the Penn Treebank) takes over 8 hours on a 1.4GHz Intel Core i5 Macbook Air with a perfect oracle providing the 1-best overt atomic tag, but just over 6 minutes using reified supertags.

5 Conclusion

We presented two methods for constraining the parser's search space and improving efficiency during wide-coverage MG parsing. The first extends the formalism with mechanisms for enforcing morphosyntactic agreements and selectional restrictions. The second anchors computationally costly null heads to overt heads inside complex overt categories, rendering the formalism fully compatible with Markovian supertagging techniques. Both techniques have proven useful for the generation of MGbank. We are now working on an A* MG parser which can consider the full distribution of supertags for each word and exploit the potential of these rich lexical categories.

Acknowledgments

A big thank you is due to Mark Steedman, as well as to the four anonymous reviewers for all their helpful comments and suggestions. Most especially, I would like to thank Miloš Stanojević, who coded up the LSTM supertagger, ran the experiments reported in this paper and made some very helpful suggestions regarding the supertagging method described above. This project was supported by the Engineering and Physical Sciences Research Council (EPSRC) and an ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a Google Faculty Award.

References

Mark C Baker. 1988. *Incorporation: A theory of grammatical function changing*. University of Chicago Press.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265.

Cedric Boeckx, Norbert Hornstein, and Jairo Nunes. 2010. *Control as Movement*. Cambridge University Press, Cambridge, UK.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1240. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

Noam Chomsky. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels, and Juan Uriagereka, editors, *Step by Step: Essays in Minimalist Syntax in Honor of Howard Lasnik*, pages 89–155. MIT Press, Cambridge, MA.

Noam Chomsky. 2001. Derivation by phase. *Ken Hale: A life in language*, pages 1–52.

Noam Chomsky. 2008. On phases. In Robert Freidin, Carlos Peregrin Otero, and Maria Zubizarreta, editors, *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, pages 133–166. MIT Press.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid. ACL.

B. Cramer and Y. Zhang. 2010. Constraining robust constructions for broad-coverage parsing with precision grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 223–231, Beijing.

Meaghan Fowlie and Alexander Koller. 2017. Parsing minimalist languages with interpreted regular tree grammars. In *Proceedings of the Thirteenth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+13)*, pages 11–20.

Jane Grimshaw. 1991. Extended projection. Unpublished manuscript, Brandeis University, Waltham, Mass. (Also appeared in J. Grimshaw (2005), Words and Structure, Stanford: CSLI).

Hendrik Harkema. 2001. *Parsing Minimalist Languages*. Ph.D. thesis, UCLA, Los Angeles, California.

Norbert Hornstein, Jairo Nunes, and Kleanthes Grohmann. 2005. *Understanding Minimalism*. Cambridge University Press.

Jungo Kasai, Bob Frank, Tom McCoy, Owen Rambow, and Alexis Nasr. 2017. Tag parsing with neural networks and vector representations of supertags. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Gregory M. Kobele. 2005. Features moving madly: A formal perspective on feature percolation in the minimalist program. *Research on Language and Computation*, 3(4):391–410.

Hilda Koopman and Dominique Sportiche. 1991. The position of subjects. *Lingua*, 85(2-3):211–258.

Richard Larson. 1988. On the double object construction. *Linguistic Inquiry*, 19:335–392.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Mike Lewis and Mark Steedman. 2014. A* ccg parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Ryan McDonald and Fernando Pereira. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. University of Pennsylvania.

Jens Michaelis. 1998. Derivational minimalism is mildly context–sensitive. In *International Conference on Logical Aspects of Computational Linguistics*, pages 179–198. Springer.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gomez-Rodriguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 833–841. Association for Computational Linguistics.

David Pesetsky. 1991. Zero syntax: Vol. 2: Infinitives. Unpublished MS., MIT.

Carl Pollard and Ivan Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA.

Andrew Radford. 2004. *Minimalist Syntax: Exploring the Structure of English*. Cambridge University Press.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore. ACL.

Ian G Roberts. 2010. *Agreement and head movement: Clitics, incorporation, and defective goals*, volume 59 of *Linguistic Inquiry Monograph*. MIT Press.

Yves Schabes, Anne Abeille, and Aravind K Joshi. 1988. Parsing strategies with 'lexicalized' grammars: application to tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 578–583. Association for Computational Linguistics.

Edward Stabler. 1997. Derivational minimalism. In *Logical Aspects of Computational Linguistics (LACL'96)*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95, New York. Springer.

Edward P. Stabler. 2001. Recognizing head movement. In *Logical Aspects of Computational Linguistics: 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings.*, volume 4, pages 245–260.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In Robert Borsley and Kirsti Börjars, editors, *Non-Transformational Syntax: A Guide to Current Models*, pages 181–224. Blackwell, Oxford.

John Torr. 2017. Autobank: a semi-automatic annotation tool for developing deep minimalist grammar treebanks. In *Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017*, pages 81–86.

John Torr and Edward P. Stabler. 2016. Coordination in minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the Twelfth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+12)*, pages 1–17.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2017. A dynamic window neural network for ccg supertagging. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3337–3343.

Wenduan Xu. 2016. Lstm shift-reduce ccg parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1764.