Lexicalised LFG[*]

Jamie Y. Findlay
jamie.findlay@ling-phil.ox.ac.uk

SE-LFG25
SOAS
26 May 2018

1 Lexicalised grammars

1.1 Elementary structures and combining operations

- A grammar can be seen as consisting of two kinds of objects: ELEMENTARY STRUCTURES and COMBINING OPERATIONS (Schabes et al. 1988a; Abeillé & Rambow 2000).
- Elementary structures can be associated with a lexical item or not. Context-free grammars (CFGs) commonly contain both types:

(1) a. S $\rightarrow$ NP VP
    b. N $\rightarrow$ *Alex*

- When *all* of the elementary structures are associated with a lexical item, we say the grammar is LEXICALISED (Schabes et al. 1988a). This is the case in Lexicalised Tree-Adjoining Grammar (LTAG: Schabes et al. 1988a; Bleam et al. 2001), for instance.

1.2 Advantages

1.2.1 Parsing

- One crucial advantage of lexicalised grammars is that they simplify parsing.
- There are a finite number of words in the string; each word in the string is associated with a finite number of elementary structures, which can only be combined in finitely many ways; therefore the number of analyses for a given sentence is finite.
- This avoids the non-termination problem which recursion often poses for parsing algorithms (Schabes et al. 1988a: 581–582).

[*]This is based on work from my DPhil thesis (Findlay in prep.); my thanks to my supervisors, Ash Asudeh and Mary Dalrymple, for their continued support.

- Lexicalised grammars have a number of other pleasing properties with respect to parsing – see Schabes (1990: Ch4) and Joshi & Schabes (1997) for details.

1.2.2 Lexicalism

- The second reason to prefer a lexicalised grammar is that it very strongly instantiates (one understanding of) lexicalism.
- Lexicalism (broad view): the lexicon is placed centre stage in the grammar, treated as a "richly structured" object, and the overall syntactic theory assumes "an articulated theory of complex lexical structure" (Dalrymple 2001: 3).[1]
- This is in contrast to a theory of the lexicon as a mere "collection of the lawless" (Di Sciullo & Williams 1987: 4), where it is simply a repository of exceptions, "incredibly boring by its very nature", about which "there neither can nor should be a theory" (*ibid.*: 3–4).
- A focus on the lexicon as the source of grammatical complexity – and in a lexicalised grammar, the lexicon essentially *is* the grammar:

> The 'grammar' consists of a lexicon where each lexical item is associated with a finite number of structures for which that item is the head. There are no separate grammar rules. There are, of course, 'rules' which tell us how these structures are composed. (Schabes et al. 1988b: 1)

> There are no rules as such in the grammar. The composition operations are universal in the sense that they are the same for all grammars in the class of lexicalized grammars. (Joshi & Schabes 1997: 97)

- Cf. also similar conclusions reached (for different reasons) in the Minimalist Program (Chomsky 1995, *et seq.*), where the syntactic component is reduced to Merge.
- A lexicalist syntactic theory already needs a richly detailed lexicon. In the most parsimonious theory, this is *all* it would need.

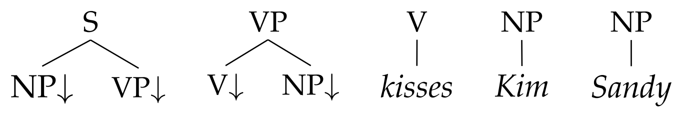
2 How to lexicalise a CFG

- LFG, being based on a CFG, is in general not lexicalised. How would we go about changing this?

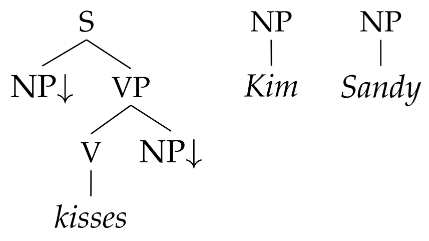
[1]What we might call the 'narrow' lexicalist view is one which adheres strictly to the LEXICALIST HYPOTHESIS (Chomsky 1970), usually encapsulated in the principle of LEXICAL INTEGRITY (Simpson 1983; Bresnan & Mchombo 1995), which states that morphological processes are distinct from syntactic ones, so that syntax is blind to the internal structure of words. This position is more contentious than perhaps it has been in the past: it has, for instance, been recently criticised as "both wrong and superfluous" by Bruening (2018). See Müller (2018) for an initial response to this, and Asudeh et al. (2013: 4–5) for some formal language arguments in favour of the lexicalist hypothesis which I think Bruening fails to address.

- We can speak of one grammar LEXICALISING another if the former is lexicalised while the latter is not, and if the two are strongly equivalent – that is, generate the same string language *and* tree language.
- CFGs cannot in general lexicalise CFGs.[2]
- We can see a CFG as a tree rewriting system where all the trees are of depth 1 (cf. McCawley's (1968) view of PSRs as 'node admissibility conditions').

(2) a. S → NP VP
  b. VP → V NP
  c. V → *kisses*
  d. NP → *Kim*
  e. NP → *Sandy*

(3)


- In general, such TREE SUBSTITUTION GRAMMARS (TSGs) need not be restricted to elementary structures of depth 1. If we give them an EXTENDED DOMAIN OF LOCALITY, we can lexicalise this CFG:

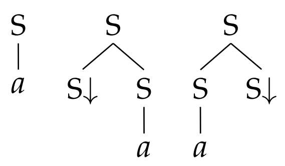
(4)


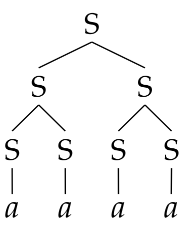
- But this approach isn't enough in general. And even in the cases where a TSG can lexicalise a CFG, the results can be linguistically unsatisfying.
- For example, the artificial language $\{a^n | n > 0\}$, that is, any number of *a*s except zero, can be straightforwardly described by the following CFG (Schabes et al. 1988a: 3; Kallmeyer 2010: 22–23):

(5) S → S S
  S → *a*

- But this *cannot* be lexicalised by a TSG, since in a TSG the distance between two nodes in an elementary tree can never grow (Schabes et al. 1988b: 3).
- (6) might be thought to lexicalise (5), but it can't produce (7):

[2]Although there is a way of converting any CFG into so-called GREIBACH NORMAL FORM (Greibach 1965), where the right-hand side of each rule begins with a terminal symbol – thereby lexicalising the grammar – such grammars do not in general generate the same set of trees as the grammars they normalise, since they will include different (and many more) rules. This means they are only weakly, not strongly, equivalent.

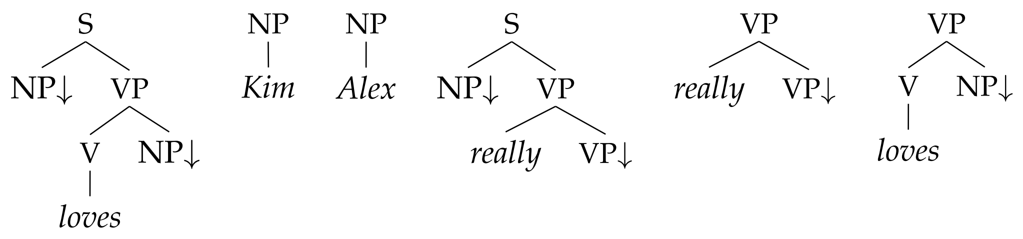
(6)


(7)


- Even when it is formally possible to lexicalise a CFG with a TSG, the results may be less than linguistically satisfying. Consider the CFG in (8):

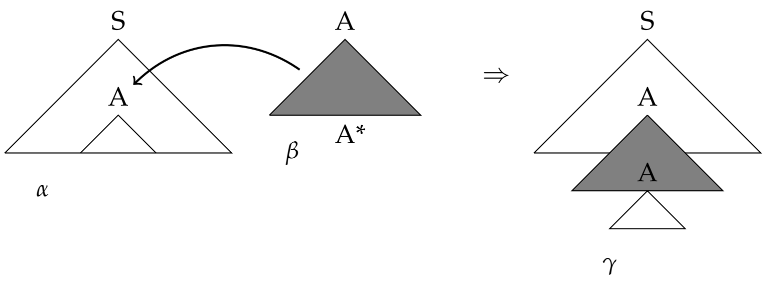
(8) $\text{S} \rightarrow \text{NP VP}$
$\text{VP} \rightarrow \text{V NP}$
$\text{NP} \rightarrow$ *Alex*
$\text{NP} \rightarrow$ *Kim*
$\text{VP} \rightarrow$ *really* VP
$\text{V} \rightarrow$ *loves*

- As Schabes et al. (1988a: 579) point out, a grammar like this can be lexicalised with the following TSG:

(9)


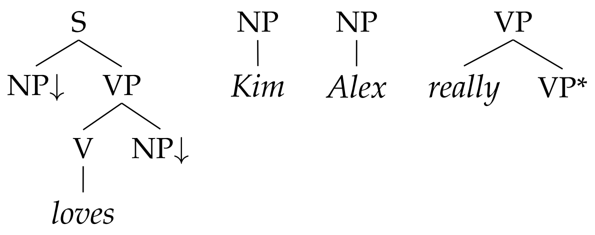
- The fourth tree is anomalous: we are forced to assume that the adverb anchors a sentential tree.

- A Tree-Adjoining Grammar (TAG: Joshi et al. 1975) avoids this undesirable consequence by using a different combining operation. Instead of substitution we have adjunction:

(10)


- This allows us to expand a tree from the inside out, thus increasing the distance between two nodes in an elementary tree (solving the first problem), and also allowing us a lot more freedom in choosing what lexical items anchor our elementary trees (solving the second). The problematic CFG can be lexicalised by the following TAG:

(11)



- Lexicalising a CFG thus 'naturally' leads to a TAG. Further grist to the mill for someone like me, . . .

3 Multiword expressions

- I have elsewhere argued that a TAG offers the best way of capturing facts about multiword expressions (MWEs) such as idioms (Findlay 2017a,b).
- The problem is that it is difficult to describe the dependencies between idiom parts at other levels of structure, so we should encode them in multiword lexical entries, which then necessitate larger syntactic objects entering into the lexicon.

4 TAG in LFG

- In Findlay (2017a,b), I proposed to associate lexical entries with (sets of) trees, instead of merely pre-terminal nodes.
- But by associating entries with trees directly, I obscured an important distinction between structures and *descriptions* of structures.
- Two kinds of grammatical theory: those which build or manipulate structures, and those which state descriptions of what structures are admissible. LFG firmly in the second camp: Kaplan (1995: 11) calls the descriptive approach the "hallmark of LFG".
- So the kinds of objects we want to associate with lexical entries are *descriptions* of trees

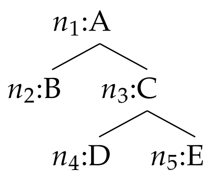
4.1 Describing trees

- A formal language for describing c-structures (based on Kaplan 1995: 10):

(12) N: set of variables over nodes
C: set of category labels, T: set of terminal labels
L: $C \cup T$
M: $N \rightarrow N$
$< \subseteq N \times N$
λ: $N \rightarrow L$

- In words:
  - a set of node variables (we can give them mnemonic names such as *vp*, *np2*, etc., or more prosaic names such as $n_1, n_2, \ldots, n_i$)
  - a set of category (non-terminal) labels (S, NP, V, etc.)
  - a set of terminal labels (*loves*, *Kim*, *and*, etc.)
  - an overarching set of labels, consisting of both the terminals and the non-terminals/categories
  - three relations – two between nodes, one between nodes and labels:
    - ▷ the (partial) function M, which maps a node to its mother
    - ▷ a partial ordering between nodes, $<$, which indicates linear precedence; it is partial because nodes only precede or follow nodes of the same 'generation'
    - ▷ the labelling function λ, which maps nodes to their labels
- Using this language, we can then describe the tree in (13) as in (14) (Kaplan 1995: 12):

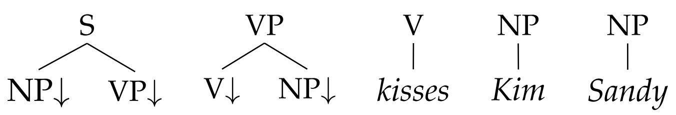
(13)



(14)
$$\begin{array}{lll} M(n_2) = n_1 & \lambda(n_1) = A & n_2 < n_3 \\ M(n_3) = n_1 & \lambda(n_2) = B & n_4 < n_5 \\ M(n_4) = n_3 & \lambda(n_3) = C & \\ M(n_5) = n_3 & \lambda(n_4) = D & \\ & \lambda(n_5) = E & \end{array}$$

- We can now represent a simple TSG as a set of tree descriptions. Substitution involves combining descriptions and equating two nodes.

(15)



(16)

a. $\mathrm{M}(\mathit{subj\text{-}np}) = s$
$\mathrm{M}(vp) = s$
$\lambda(s) = \mathrm{S}$
$\lambda(\mathit{subj\text{-}np}) = \mathrm{NP}$
$\lambda(vp) = \mathrm{VP}$
$\mathit{subj\text{-}np} < vp$

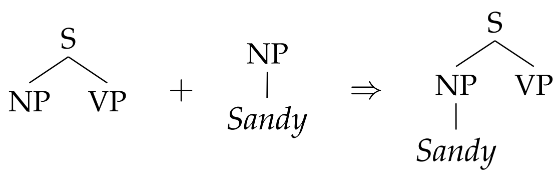
b. $\mathrm{M}(v) = vp$
$\mathrm{M}(np) = vp$
$\lambda(vp) = \mathrm{VP}$
$\lambda(v) = \mathrm{V}$
$\lambda(np) = \mathrm{NP}$
$v < np$

c. $\mathrm{M}(\mathit{kisses}) = v$
$\lambda(v) = \mathrm{V}$
$\lambda(\mathit{kisses}) = \mathit{kisses}$

d. $\mathrm{M}(\mathit{kim}) = np$
$\lambda(np) = \mathrm{NP}$
$\lambda(\mathit{kim}) = \mathit{Kim}$

e. $\mathrm{M}(\mathit{sandy}) = np$
$\lambda(np) = \mathrm{NP}$
$\lambda(\mathit{sandy}) = \mathit{Sandy}$

(17)



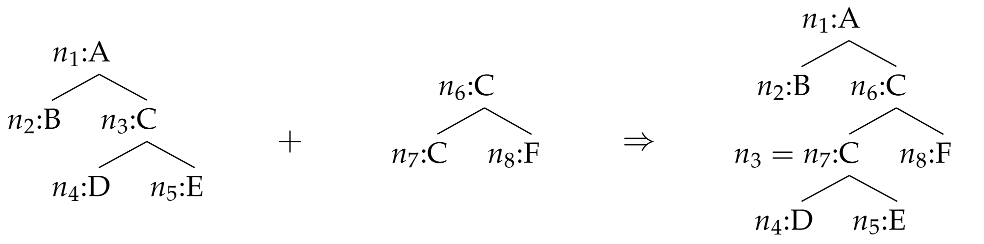
(18)

$\mathrm{M}(\mathit{subj\text{-}np}) = s$	$+$	$\mathrm{M}(\mathit{sandy}) = np$	$\Rightarrow$	$\mathrm{M}(\mathit{subj\text{-}np}) = s$
$\mathrm{M}(vp) = s$		$\lambda(np) = \mathrm{NP}$		$\mathrm{M}(vp) = s$
$\lambda(s) = \mathrm{S}$		$\lambda(\mathit{sandy}) = \mathit{Sandy}$		$\mathrm{M}(\mathit{sandy}) = np$
$\lambda(\mathit{subj\text{-}np}) = \mathrm{NP}$				$\lambda(s) = \mathrm{S}$
$\lambda(vp) = \mathrm{VP}$				$\lambda(\mathit{subj\text{-}np}) = \mathrm{NP}$
$\mathit{subj\text{-}np} < vp$				$\lambda(vp) = \mathrm{VP}$
				$\lambda(np) = \mathrm{NP}$
				$\lambda(\mathit{sandy}) = \mathit{Sandy}$
				$\mathit{subj\text{-}np} < vp$
				$\boxed{\mathit{subj\text{-}np} = np}$

4.2 Description-based TAG

- Fully specified descriptions won't work for a TAG, since adjunction changes the relationships between nodes (Vijay-Shanker 1992).

(19)

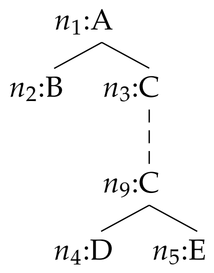


$\mathrm{M}(n_3) = n_1$ $\qquad\qquad$ $\mathrm{M}(n_3) = n_6 \neq n_1$

- Instead, we have to describe potential adjunction sites in terms of two QUASI-NODES – nodes which may or may not turn out to be the same, and which, if they are different, may appear arbitrarily far apart (but one must be the ancestor of the other).

- We can represent this using a dashed line (Vijay-Shanker 1992: 488):

(20)



- This relation is just simple dominance, rather than *immediate* dominance, which is what the M function describes. We define dominance as the transitive, reflexive closure of the daughter relation (itself the inverse of the mother function):

(21) $Dom := \{\langle x, y\rangle \mid \mathrm{M}^{-1^*}(x) = y\}$

- (Basically, a node dominates itself and any of its descendants.)

(22)

$$\begin{array}{l} \qquad\vdots \\ \mathrm{M}(n_3) = n_1 \\ \mathrm{M}(n_4) = n_9 \\ \mathrm{M}(n_5) = n_9 \\ Dom(n_3, n_9) \\ \qquad\vdots \end{array}$$

- It *could* end up that $\mathrm{M}(n_9) = n_1$ if it so happens that $n_9 = n_3$. And this is exactly what we assume when there is no adjunction, since we want the *minimal* structure which satisfies the description.

- Once again we can reduce the combining operation to simple concatenation and unification (with a few linguistically-motivated constraints, such as that frontier nodes have to be terminals, etc.).

5 Lexical entries

- Traditional LFG lexical entries are formally just CFG phrase structure rules that have only a terminal symbol on the right hand side:

(23)

$$\mathrm{N} \;\rightarrow\; \begin{array}{c} \textit{cat} \\ (\uparrow \textsc{pred}) = \text{'cat'} \\ (\uparrow \textsc{num}) = \textsc{sg} \\ (\uparrow \textsc{pers}) = 3 \end{array}$$

- Often also represented as a triple (W, C, F):

(24) *cat* N $(\uparrow \textsc{pred}) = \text{‘cat’}$
$(\uparrow \textsc{num}) = \textsc{sg}$
$(\uparrow \textsc{pers}) = 3$

- This makes clear their somewhat hybrid nature: W and C give c-structure information; F describes all other levels of structure.
- If we treat c-structure information as a description too, we can reduce all lexical entries to nothing but descriptions, i.e. just F.
- This way of looking things doesn't require a TAG – as we saw, we can conceive of CFG PSRs as descriptions too. But a lexicalised grammar does allow us to reduce the grammar to *just* lexical entries (and lexicalisation of a CFG can be achieved by a TAG).

6 A template-based meta-grammar

- One advantage of seeing c-structure information as descriptions is that we can abbreviate descriptions in TEMPLATES (Dalrymple et al. 2004; Crouch et al. 2012).
- Firstly, this makes lexical entries more readable.
- Secondly, this network of templates will have its own hierarchical structure, which is itself an interesting object of investigation, and allows us to express relationships more readily than via PSRs.
- There is, for instance, no stated relationship between the two rules in (25), even though the latter is obviously partially described by the former:

(25) a. VP $\rightarrow$ V NP
b. VP $\rightarrow$ V NP NP

- But if we have a DITRANSITIVE template which itself calls a TRANSITIVE template, the inheritance relationship is made explicit.

(26) a. TRANSITIVE :=
$\text{M}(\textit{trans-v}) = \textit{trans-vp}$
$\text{M}(\textit{trans-np}) = \textit{trans-vp}$
$\lambda(\textit{trans-vp}) = \text{VP}$
$\lambda(\textit{trans-v}) = \text{V}$
$\lambda(\textit{trans-np}) = \text{NP}$
$\textit{trans-v} < \textit{trans-np}$

b. DITRANSITIVE :=
@TRANSITIVE
$\text{M}(\textit{ditrans-np}) = \textit{trans-vp}$
$\lambda(\textit{ditrans-np}) = \text{NP}$
$\textit{trans-np} < \textit{ditrans-np}$

*** **Aside: template logic** ***

- In the standard template logic, it is not clear that anything guarantees that the same variable name in two different templates in the same lexical entry refers to the same thing.
- Suggestion: we enforce an ordering between the expansion of templates and the instantiation of variables so that the former precedes the latter. In other words, we insist that the grammar be fully compiled before variables are instantiated.
- Templates offer a way of capturing commonalities and generalisations across multiple objects in the grammar, but they are ultimately just abbreviations for the functional descriptions they contain (i.e. a grammar with templates is extensionally equivalent to one without – see Asudeh et al. (2013: 17ff.) for discussion).
- The 'real' grammar is the fully spelt out result of expanding all templates in all lexical entries (and phrase structure rules in a grammar which contains them).
- If we enforce the ordering expansion $<$ instantiation, all identically-spelled variables in the same description will in fact be instances of the same variable, unlike if we allow variables to be instantiated before unpacking templates.
- To see the difference, consider a simple mathematical example involving two templates:

(27) TEMPLATE-A :=
$x + 3 = 5$

(28) TEMPLATE-B :=
@TEMPLATE-A
$x^2 = 4$

- Instantiation $<$ expansion:

(29) $2 + 3 = 5$
$x^2 = 4$

Solution: $x = \pm 2$

- Expansion $<$ instantiation:

(30) $x + 3 = 5$
$x^2 = 4$

Solution: $x = +2$.

- Potential issue: Any templates that might be reused in a lexical entry have to be parametrised (unless we want any variables they introduce to refer to the same thing).

- For instance, if we wanted a template NP-Node which introduced an NP node, it would have to be (31), not (32):

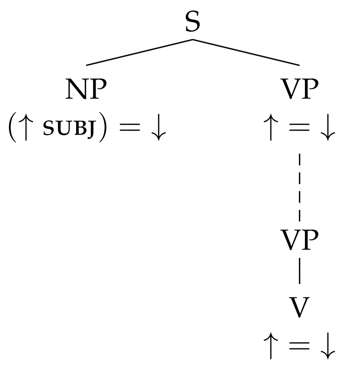
(31) NP-Node(X) :=
$$\lambda(X) = \text{NP}$$

(32) NP-Node :=
$$\lambda(np) = \text{NP}$$

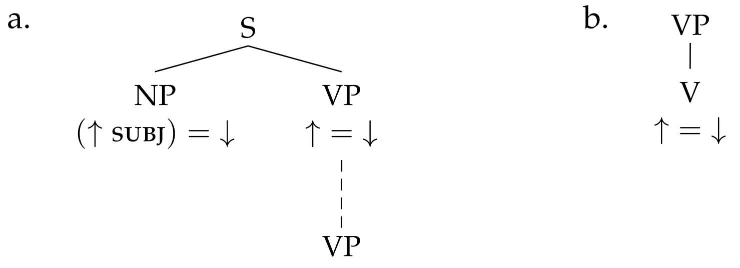
- Do we need to parametrise all of the c-structure templates ... ?
- (What does XLE do? Usually the only variables mentioned in descriptions are the node variables $*$ and $\hat{*}$, but their value is always determinate: they will only ever be instantiated on a particular node, and that then fixes their value, so it is irrelevant whether they are instantiated before or after template expansion. They can also never refer to the same node.)

*** **Aside ends** ***

- As an example, let us look briefly at intransitive verbs. These require the following annotated quasi-tree for active voice declarative sentences:

(33)


- Many other kinds of verb share the first part of this tree, the subject NP position, so we can factor out the two parts, and assign them to templates:

(34)


(35) a. Canonical-Subject :=

$M(\textit{canon-subj-np}) = \textit{canon-subj-s}$
$M(\textit{canon-subj-vp1}) = \textit{canon-subj-s}$
$Dom(\textit{canon-subj-vp1}, \textit{canon-subj-vp2})$
$\lambda(\textit{canon-subj-s}) = \mathrm{S}$
$\lambda(\textit{canon-subj-np}) = \mathrm{NP}$
$\lambda(\textit{canon-subj-vp1}) = \mathrm{VP}$
$\lambda(\textit{canon-subj-vp2}) = \mathrm{VP}$
$\textit{canon-subj-np} < \textit{canon-subj-vp1}$
$(\phi(\textit{canon-subj-s})\ \textsc{subj}) = \phi(\textit{canon-subj-np})$
$\phi(\textit{canon-subj-s}) = \phi(\textit{canon-subj-vp1})$

b. Basic-VP := [to be revised]

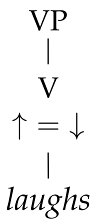
$M(\textit{basic-VP-v}) = \textit{basic-VP-vp}$
$\lambda(\textit{basic-VP-vp}) = \mathrm{VP}$
$\lambda(\textit{basic-VP-v}) = \mathrm{V}$
$\phi(\textit{basic-VP-vp}) = \phi(\textit{basic-vp-v})$

- We then parametrise the second template to allow it to take a lexical head:

(36) Basic-VP(X) :=

$M(\textit{basic-VP-v}) = \textit{basic-VP-vp}$
$M(\textit{basic-VP-head}) = \mathrm{V}$
$\lambda(\textit{basic-VP-vp}) = \mathrm{VP}$
$\lambda(\textit{basic-VP-v}) = \mathrm{V}$
$\phi(\textit{basic-VP-vp}) = \phi(\textit{basic-vp-v})$
$\lambda(\textit{basic-VP-head}) = \mathrm{X}$

- For instance, a call of @Basic-VP(*laughs*) describes the following tree:

(37) 

- Of course, the canonical subject position is not the only possible way to realise the subject of an intransitive verb. There is also the possibility of questioning it, and of relativising on it:

(38) a. Who laughs when they're in pain?
b. Someone who laughs all the time is a pleasure to be around.

- We can put all of these together into a single Subject template:

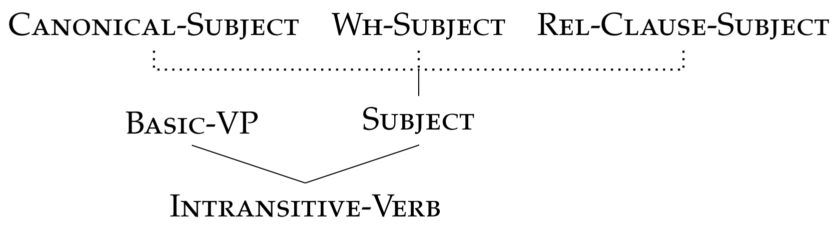
(39) Subject :=
{@Canonical-Subject | @Wh-Subject | @Rel-Clause-Subject | … }

(40) Intransitive-Verb(X) :=
@Basic-VP(X)
@Subject

- The following example gives the lexical entry for *laughs*, making use of the c-structure template and including the usual f-structure information in the new format:

(41) **laughs:**
@Intransitive-Verb(*laughs*)
(ϕ(*basic-VP-v*) pred) = 'laugh'
(ϕ(*basic-VP-v*) tense) = pres
(ϕ(*basic-VP-v*) subj num) = sg
(ϕ(*basic-VP-v*) subj pers) = 3

- The relationships between templates can be captured in a hierarchy, similar to the inheritance hierachies familiar from HPSG and other formalisms (but with crucial differences – see Asudeh et al. (2013: 17–20) for further discussion):

(42)


- This is a kind of meta-grammar (Candito 1996; Crabbé et al. 2013), which captures generalisations across all levels of structure.

- Using templates like this might also enable cross-linguistic generalisations to be captured more perspicuously (e.g. we could factor out dominance from linear order in the style of ID/LP grammars).

7 Summary

- **Goal:** integrate a TAG into the LFG architecture.

- (**Motivation 1:** satisfactory description of MWEs.)

- **Motivation 2:** a lexicalised grammar.

- **Bonus 1:** lexical entries just contain descriptions, rather than descriptions plus miscellaneous c-structure information. We reintegrate c-structure into the description-based approach which is the hallmark of LFG.

- **Bonus 2:** a different/potentially more interesting way of talking about c-structure in LFG.

References

Abeillé, Anne & Owen Rambow. 2000. Tree Adjoining Grammar: an overview. In Anne Abeillé & Owen Rambow (eds.), *Tree Adjoining Grammars: Formalisms, linguistic analysis and processing*, Stanford, CA: CSLI Publications.

Asudeh, Ash, Mary Dalrymple & Ida Toivonen. 2013. Constructions with lexical integrity. *Journal of Language Modelling* 1(1). 1–54. `http://jlm.ipipan.waw.pl/index.php/JLM/article/view/56/49`.

Bleam, Tonia, Chung-hye Han, Rashmi Prasad, Carlos Prolo & Anoop Sarkar. 2001. A Lexicalized Tree Adjoining Grammar for English. Tech. rep. from the Institute for Research in Cognitive Science, University of Pennsylvania. `http://www.cis.upenn.edu/~xtag/`.

Bresnan, Joan & Sam A. Mchombo. 1995. The lexical integrity principle: evidence from Bantu. *Natural Language and Linguistic Theory* 13(2). 181–254. `http://doi.org/10.1007/BF00992782`.

Bruening, Benjamin. 2018. The lexicalist hypothesis: both wrong and superfluous. *Language* 94(1). 1–42. `https://doi.org/10.1353/lan.2018.0000`.

Candito, Marie-Hélène. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th conference on Computational Linguistics (COLING)*, 194–199. Association for Computational Linguistics. `http://dx.doi.org/10.3115/992628.992664`.

Chomsky, Noam. 1970. Remarks on nominalization. In Roderick A. Jacobs & Peter S. Rosenbaum (eds.), *Readings in English transformational grammar*, 184–221. Waltham, MA: Ginn.

Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, UK: Cambridge University Press.

Crabbé, Benoît, Denys Duchier, Claire Gardent, Joseph Le Roux & Yannick Parmentier. 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics* 39(3). 591–629.

Crouch, Dick, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell & Paula Newman. 2012. *XLE documentation*. Palo Alto Research Center (PARC), Palo Alto, CA. `http://www2.parc.com/isl/groups/nltt/xle/doc/xle.html`.

Dalrymple, Mary. 2001. *Lexical Functional Grammar* (Syntax and Semantics 34). Stanford, CA: Academic Press.

Dalrymple, Mary, Ronald M. Kaplan & Tracy Holloway King. 2004. Linguistic generalizations over descriptions. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG04 Conference*, 199–208. Stanford, CA: CSLI Publications. `http://www.stanford.edu/group/cslipublications/cslipublications/LFG/9/lfg04dkk.pdf`.

Di Sciullo, Anna Maria & Edwin Williams. 1987. *On the definition of word* (Linguistic Inquiry monographs 14). Cambridge, MA: MIT Press.

Findlay, Jamie Y. 2017a. Multiword expressions and lexicalism. In Miriam Butt &

Tracy Holloway King (eds.), *Proceedings of the LFG17 Conference*, 209–229. Stanford, CA: CSLI Publications. http://web.stanford.edu/group/cslipublications/cslipublications/LFG/LFG-2017/lfg2017-findlay.pdf.

Findlay, Jamie Y. 2017b. Multiword expressions and lexicalism: the view from LFG. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, 73–79. Association for Computational Linguistics. http://aclweb.org/anthology/W17-1709.

Findlay, Jamie Y. in prep. Multiword expressions and the lexicon. DPhil thesis, University of Oxford.

Greibach, Sheila A. 1965. A new normal-form theorem for context-free phrase structure grammars. *Journal of the Association for Computing Machinery (JACM)* 12(1). 42–52. http://doi.acm.org/10.1145/321250.321254.

Joshi, Aravind K., Leon S. Levy & Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10(1). 136–163.

Joshi, Aravind K. & Yves Schabes. 1997. Tree-Adjoining Grammars. In Grzegorz Rozenberg & Arto Salomaa (eds.), *Handbook of formal languages*, vol. 3, 69–123. Berlin, DE: Springer.

Kallmeyer, Laura. 2010. *Parsing beyond context-free grammars*. Heidelberg, DE: Springer.

Kaplan, Ronald M. 1995. The formal architecture of Lexical-Functional Grammar. In Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell, III, & Annie Zaenen (eds.), *Formal issues in Lexical-Functional Grammar*, 7–28. Stanford, CA: CSLI Publications.

McCawley, James D. 1968. Concerning the base component of a transformational grammar. *Foundations of Language* 4(3). 243–269.

Müller, Stefan. 2018. The end of lexicalism as we know it? *Language* 94(1). e54–e66. https://doi.org/10.1353/lan.2018.0014.

Schabes, Yves. 1990. Mathematical and computational aspects of lexicalized grammars. Doctoral dissertation, University of Pennsylvania.

Schabes, Yves, Anne Abeillé & Aravind K. Joshi. 1988a. Parsing strategies with 'lexicalized' grammars: application to Tree Adjoining Grammars. In *Proceedings of the 12th Conference on Computational Linguistics (COLING)*, 578–583. Association for Computational Linguistics. http://dx.doi.org/10.3115/991719.991757.

Schabes, Yves, Anne Abeillé & Aravind K. Joshi. 1988b. Parsing strategies with 'lexicalized' grammars: application to Tree Adjoining Grammars [revised version]. Tech. Rep. MS-CIS-88-65, University of Pennsylvania Department of Computer and Information Science.

Simpson, Jane. 1983. Aspects of Warlpiri morphology and syntax. Doctoral dissertation, MIT.

Vijay-Shanker, K. 1992. Using descriptions of trees in a Tree Adjoining Grammar. *Computational Linguistics* 18(4). 481–517.