Parsing as Deduction Revisited: Using an Automatic Theorem Prover to Solve an SMT Model of a Minimalist Parser

Sagar Indurkhya
Massachusetts Institute of Technology
32 Vassar St.
Cambridge, MA 02139
indurks@mit.edu

Abstract

We introduce a constraint-based parser for Minimalist Grammars (MG), implemented as a working computer program, that falls within the long established "Parsing as Deduction" framework. The parser takes as input an MG lexicon and a (partially specified) pairing of sound with meaning – i.e. a word sequence paired with a semantic representation – and, using an axiomatized logic, declaratively deduces syntactic derivations (i.e. parse trees) that comport with the specified interface conditions. The parser is built on the first axiomatization of MGs to use Satisfiability Modulo Theories (SMT), encoding in a constraint-based way the principles of minimalist syntax. The parser operates via a novel solution method: it assembles an SMT model of an MG derivation, translates the inputs into SMT formulae that constrain the model, and then solves the model using the Z3 SMT-solver, a high-performance automatic theorem prover; as the SMT-model has finite size (being bounded by the inputs), it is decidable and thus solvable in finite time. The output derivation is then recovered from the model solution. To demonstrate this, we run the parser on several representative inputs and examine how the output derivations differ when the inputs are partially vs. fully specified. We conclude by discussing the parser's extensibility and how a linguist can use it to automatically identify: (i) dependencies between input interface conditions and principles of syntax, and (ii) contradictions or redundancies between the model axioms encoding principles of syntax.

1 Introduction

Minimalist theories of syntax consider the *Human Language Faculty* (HLF) as a computational system capable of deriving from a finite lexicon and a single combinatorial operation, an unbounded set of hierarchical syntactic structures, pairing sounds (typically word sequences) with meaning representations (Chomsky, 1995). (In more technical language, the HLF pairs *Phonological Forms* [PF], where a PF is an encoding of information relevant to how a brain-internal structured expression gets pronounced, signed, etc, with *Logical Forms* [LF], where an LF is a structured semantic representation, e.g. predicate-argument structure.) This study introduces a novel computational model for the HLF, implemented as a working computer program,[1] that takes the form of a constraint-based parser for Minimalist Grammars (MG), grounded in the (first) axiomatization of minimalist syntax using Satisfiability Modulo Theories (SMT).[2] Working within the *"Parsing as Deduction"* framework (Pereira and Warren, 1983), the parser is a logic program that uses an automatic theorem prover to answer the question: *can a given lexicon yield a syntactic structure that encodes a given LF and/or PF?*

More specifically, the parser takes as input an MG lexicon and a (partial) specification of LF and PF interface conditions (i.e. constraints over the LF and PF encoded in a syntactic structure), and it outputs the set of MG derivations (i.e. syntactic structures) that the (input) lexicon can generate and that satisfy the (input) interface conditions. The parser operates by first constructing an SMT model of a lexicon and an SMT model of derivation, with the two models linked by shared free variables to form an SMT model of a minimalist parser. Next, the parser converts the inputs into constraints, expressed as SMT-formulae, that augment the SMT model and serve to constrain the space of model solutions. Finally, the parser obtains its output by using the Z3 SMT-solver,[3] a (modern) high-performance automatic theorem

[1] The program's source code is available at https://github.com/indurks/mgsmt.

[2] SMT is a propositional logic that may be extended with background theories – e.g. the theories of uninterpreted functions, bit-vectors and arithmetic (Dutertre and de Moura, 2006; Ranise and Tinelli, 2006; Nieuwenhuis and Oliveras, 2006; Nieuwenhuis et al., 2006; Moura and Bjørner, 2009).

[3] See (Moura and Bjørner, 2008; Bjørner, 2011).

Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL), pages 157 - 175

prover, to check whether the SMT model is satisfiable – if it is, the SMT-solver enumerates valid model-interpretations from which the parser recovers the (output) set of minimalist derivations.

Notably, this model of HLF is declarative, and so encompasses both semantic parsing and natural language generation. E.g. one can use the parser to generate language by: (i) inputting a lexicon and LF constraints; (ii) ordering the parser to "solve for syntax" and recover a derivation from the model-solution; and (iii) obtaining the (output) generated PF from the recovered derivation. (Here the inputs are known quantities and the derivation is an unknown quantity being solved for.) Moreover, our model for HLF can be used to run experiments in which the input interface conditions are *partially* specified and the SMT-solver is instructed to identify dependencies between the principles of syntax (encoded in the parser) and the features in the input lexicon – in this way, one can determine whether (and how) the syntactic principles and the lexicon do not adequately constrain a derivation to compensate for the absent (LF or PF) interface conditions.

The remainder of this study is organized as follows. First, §2 reviews key principles of minimalist syntax and how they are modeled using MGs. Next, §3 reviews related prior work within the *Parsing as Deduction* framework, which this study seeks to extend, and that motivates our approach. Then, §4, §5 and §6 present the three key contributions of this study: §4 details the deductive parsing procedure, showing how the Z3 SMT-solver can be used to identify satisfiable interpretations of an SMT model of a minimalist parser; §5 details the SMT model of the minimalist parser, its underlying axiomatization of minimalist syntax, and how the model is constrained by user specified inputs; §6 details application of the parser to a representative set of example inputs and analyzes the output derivations, showing how the parser functions even when the input interface conditions are only partially specified. Finally, §7 discusses how: (i) the SMT model of the parser may be extended, and (ii) the parser can help linguists identify dependencies and contradictions between the model axioms encoding principles of syntax and the logical constraints derived from the input interface conditions.

2 Background: Minimalist Grammars

We opted to model minimalist syntax using the Minimalist Grammar (MG) formalism (Stabler, 1996) because MGs have been extensively characterized formally and appear to be sufficiently expressive for modeling the syntactic structures prescribed by contemporary theories of minimalist syntax.[4] The MG formalism (and minimalist syntax more generally) centers on: (i) a lexicon consisting of a finite set of lexical items (i.e. syntactic atoms), each pairing a word with a finite sequence of syntactic features, and (ii) *Merge*, a recursive, binary structure-building operation. Syntactic structures are derived from a multi-set of lexical items via repeated application of *Merge*, which has two (logically-disjoint) sub-cases, *external merge* (EM) and *internal merge* (IM),[5] that serve to model two basic facts of natural language, *combination* and *displacement* (respectively).[6]

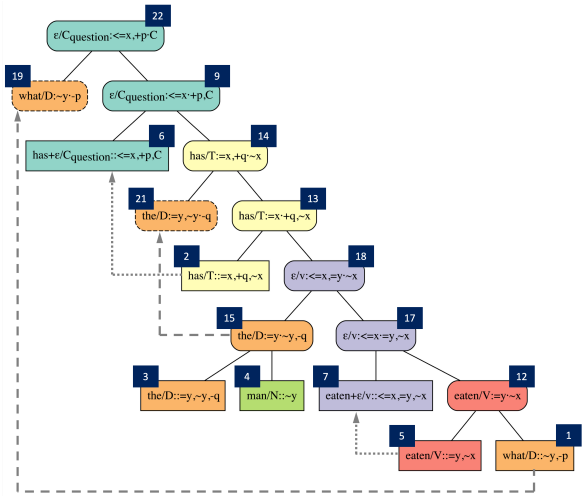

Figure 1: The parser outputs an MG derivation of *"What has the man eaten?"* that satisfies the LF & PF interface conditions in I_1 (of Table 2). The derivation was recovered from the model interpretation in Table 3, and each node is labeled with the index of a row in Table 3. The depicted structure is a multi-dominance tree, with nodes $\{1, 5, 12, 7, 17, 3, 4, 15, 18, 2, 13, 14, 6, 9, 22\}$ making up the derivation tree from which this multi-dominance tree was derived. Lexical and derived nodes are denoted by regular and rounded rectangles respectively. Constituents with the same head have the same color. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the (raised) lower structure being copied to the target position.

[4] See (Michaelis, 1998; Michaelis et al., 2000; Michaelis, 2001; Graf, 2011, 2013; Kobele, 2011).

[5] *EM* merges two disjoint structures, whereas *IM* merges a structure with one of its sub-structures.

[6] Combination forms syntactic structures by (recursively) pairing separate structures; it is used to associate predicates with their arguments (i.e. the assignment of thematic roles like "Agent" and "Patient," also known as θ-roles). Displace-

To illustrate the MG formalism, let us see how the MG derivation (i.e. syntactic structure) for the sentence *"What has the man eaten?"*, shown in Fig. 1, is built bottom-up using the lexical items listed in Table 1. First, the lexical items for the determiner *"the"* and the nominal *"man"* are combined, via the application of external merge, to form the determiner phrase *"the man"*; note that this instance of *constituent selection* is allowed because the term *"the"* has a *selector* feature, $=y$, that matches the *selectee* feature, $\sim y$, on the term *"man"*.[7] Then, the lexical items for the (lexical) verb *"eaten"* is first (externally) merged with its complement, the (internal) argument *"what"* to form a VP, which is then (externally) merged with a covert light-verb, ϵ/v, with the resulting vP then merged with the external argument, *"the man"*, to form a (double) VP-shell structure in accordance with the Hale-Keyser model of predicate-argument structure (Hale and Keyser, 1993, 2002). Next, the VP-shell structure is merged with a tense marker, the auxiliary verb *"has"*, to form a TP. After this, per the *VP Internal Subject Hypothesis* (Radford, 2009), the internal argument, *"the man"* is moved, via application of *internal merge*, from its initial location (within the VP-shell) to the subject-position of the TP; note that this instance of movement is licensed by the *licensor* feature, $+q$, on *"has"* matching the *licensee* feature, $-q$, on *"the man"*. The TP is then (externally) merged with a (covert) complementizer, ϵ/C, to form a CP.[8] Finally, the internal argument *"what"* is raised (via internal merge) from the VP-shell to the specifier position of the CP, at which point the derivation is complete.[9]

In summary, to parse a sentence, a multi-set of lexical items is selected from the lexicon and (recursively) *merged* together to yield a derivation in which the terminal expression has only the special feature C remaining (because all of the selectional and licensing features have been consumed); if the ordering of the phonological forms in the resulting structure aligns with the order of the words in the sentence being parsed,[10] then the structure is considered to be a valid parse of the sentence.[11]

3 Related Work: Parsing as Deduction

We have developed an MG parser within the *Parsing as Deduction* framework, which was first described by Pereira and Warren (1983), who showed how an axiomatization of a context-free grammar could be combined with a logical deduction engine to formulate a chart parser as a logic program. As Pereira notes, key advantages of this framework include: (i) a connection between the deductions that yield a syntactic structure and the inferences needed to extract a semantic interpretation from said structure; (ii) the ability to handle filler-gap dependencies without altering the basic design of a chart parser. The *Parsing as Deduction* framework has since been employed to construct parsers for a variety of grammatical formalisms, including lexicalized context-free grammars, tree adjoining grammars, combinatory categorical grammars, and dependency grammars.[12] Notably, this framework has been used to develop parsers that model Government and Binding (GB) theory (a predecessor of minimalist syntax) by encoding principles of syntax within a system of axioms that mirrors the modular structure of GB theory (Chomsky, 1981; Johnson, 1989; Fong, 1991).

Normally, these parsers employ Prolog, the de-facto language for Constraint Logic Programming (CLP).[13] However, we leverage recent advances in the performance of automated theorem provers for SMT, which enhances CLP by enabling us to focus entirely on formulating (declarative) model axioms while the computer is free to decide how best to deduce a model solution (De Moura and Bjørner,

ment, driven by syntactic movement, enables a phrase to be interpreted at both its (final) surfaced position as well as other positions within a syntactic structure – e.g., given the expression *"You, I love."*, *"You"* is the object of *"love"* and normally appears in *Object* position, but here it is displaced to the front of the sentence (where it is pronounced).

[7] Selector, selectee, licensor and licensee features are designated by a prefixed $=$, $\sim$, $+$, and $-$ respectively.

[8] The *extended projection*, $C{-}T{-}v{-}V$, forms the spine of each clause (Grimshaw, 2005; Adger and Svenonius, 2011).

[9] N.b. *head-movement* – i.e. the incorporation of a lower (lexical) head into the head it merges with – is applied when the completed derivation is sent to the PF-interface for externalization. Head-movement occurs twice in this derivation: (i) the V-to-v head-movement utilized in the Hale-Keyser model of predicate-argument structure; (ii) the T-to-C head-movement utilized in raising the auxiliary verb (as when forming a polar-interrogative from a declarative).

[10] E.g. using Specifier-Head-Complement linearization to model *Subject-Verb-Object* (SVO) ordering (Kayne, 1994).

[11] See Appendix-B for further commentary on MGs, including a presentation of an algebraic formulation of MGs based on (Stabler and Keenan, 2003).

[12] See (Shieber et al., 1995; Duchier, 1999; Tang and Mooney, 2001; Debusmann et al., 2004; Estratat and Henocque, 2004; Duchier et al., 2010; Lierler and Schüller, 2012; Schüller, 2013). See (Schabes and Waters, 1993; Joshi and Schabes, 1997; Steedman and Baldridge, 2011) for details of these grammatical formalisms.

[13] See (Jaffar and Lassez, 1987; Apt, 1990; Jaffar and Maher, 1994; Koller and Niehren, 2002).

1. $\epsilon/C_{Ques.} :: <=x, +p, C$	19. $he :: \sim y, -q$
2. $has :: =x, +q, \sim x$	20. $resigned :: \sim x$
3. $the :: =y, \sim y, -q$	21. $known :: =y, \sim x$
4. $man :: \sim y$	22. $everyone :: \sim y, -q, -p$
5. $\epsilon/v :: <=x, =y, \sim x$	23. $who :: =x, +p, \sim y$
6. $eaten :: =y, \sim x$	24. $loved :: =y, \sim x$
7. $what :: \sim y, -p$	25. $\epsilon/C_{Decl.} :: =x, C$
8. $\epsilon/v :: <=x, \sim x$	26. $knows :: =y, \sim x$
9. $\epsilon/C_{Ques.} :: <=x, C$	27. $john :: \sim y, -q$
10. $was :: =x, +q, \sim x$	28. $given :: =y, \sim x$
11. $she :: \sim y, -q$	29. $\epsilon/T :: =x, +q, \sim x$
12. $given :: =y, =y, \sim x$	30. $money :: \sim y, -q, -p$
13. $money :: \sim y$	31. $that :: =x, +p, \sim y$
14. $will :: =x, +q, \sim x$	32. $stolen :: =y, \sim x$
15. $who :: \sim y, -q, -p$	33. $fears :: =y, \sim x$
16. $her :: \sim y$	34. $money :: \sim y, -q$
17. $tell :: =y, =y, \sim x$	35. $\epsilon/C_{Ques.} :: =x, +p, C$
18. $that :: =x, \sim y$	36. $a :: =y, \sim y, -q$

Table 1: An MG lexicon that the parser may take as input. Each lexical item consists of: (i) a phonological form that is either overt or covert (ϵ); (ii) (optional) a categorical feature (e.g. entries 1 & 5); (iii) a sequence of syntactic features. The lexicon includes entries for auxiliary verbs (e.g. 2, 10 & 14), determiners (e.g. 3), nominals (e.g. 4, 11, 22, 27 & 30), tense markers (e.g. 2, 14, & 29), complementizers (e.g. 1, 9, 18 & 25), relative pronouns (e.g. 23), Wh-words (e.g. 7 & 15), intransitive verbs (e.g. 20), transitive verbs (e.g. 6, 26 & 32), and ditransitive verbs (e.g. 12 & 17).

2011). Hence, we extend prior work within the *Parsing as Deduction* framework by: (i) developing a (declarative) constraint-based *minimalist* parser, thereby advancing (linguistically) beyond earlier *GB*-based parsers; (ii) formulating an MG parser as a finite (and thus decidable) SMT-model that is solved using an SMT-solver (instead of Prolog).[14]

4 The Parsing Procedure

This section details the parsing procedure and illustrates it with a worked out example.

INPUT. The procedure takes as input: (i) an MG lexicon, $\mathcal{L}$; (ii) a pairing of LF and PF interface conditions, I, to be parsed; (iii) parameters, p, bounding the size of the SMT model (to be built).

INITIALIZATION. The procedure initializes the SMT-solver with an empty stack of constraints, $\mathcal{S}$.

CONSTRUCTING THE SMT MODEL. The SMT model of the parser is constructed as follows. First, the procedure instantiates the SMT model of the lexicon (detailed in §5) and constrains it with the input lexicon – this is carried out by:

(a) initializing an SMT model of a lexicon, $m_{\mathcal{L}}$, with size bound by p, and pushing $m_{\mathcal{L}}$ onto $\mathcal{S}$;

(b) constructing an SMT-formula, c_l, that restricts interpretations (i.e. model solutions) of $m_{\mathcal{L}}$ to align with $\mathcal{L}$, and then pushing c_l onto $\mathcal{S}$;

Next, the procedure instantiates an SMT model of a derivation (detailed in §5) and then constrains it with the (input) interface conditions – this involves:

(a) initializing an SMT model of a derivation, m_d, with size bound by p, and pushing m_d onto $\mathcal{S}$;

(b) translating I into an SMT-formula, c_I, that constrains m_d (detailed in §5) such that any derivation recovered from an interpretation of m_d must respect I, and pushing c_I onto $\mathcal{S}$.

Finally, the procedure "connects" the SMT model of the derivation to the SMT model of the lexicon – this is achieved by first creating an SMT-formula, m_b, that connects m_d with m_l by constraining interpretations of the free variables that appear in both m_d and m_l, and then pushing m_b onto the $\mathcal{S}$.

CHECKING THE SMT MODEL. The procedure uses the SMT-solver's model-checking routine (i.e. decision procedure) to determine whether there exists a satisfiable interpretation of the model (i.e. the conjunction of the SMT-formulae in $\mathcal{S}$) – if one exists, the procedure recovers it from the solver, and then (automatically) reconstructs an MG derivation from the (recovered) model interpretation. The procedure then pushes onto $\mathcal{S}$ a constraint (i.e. an SMT-formula) that prohibits the interpretation of m_d from being equivalent to any previously recovered (satisfiable) model interpretations;[15] this model-checking process is then run again to try and recover a (new) alternative MG derivation – this process is repeated until the solver cannot identify a (new) satisfiable model interpretation (because all model-solutions have already been identified).

OUTPUT. The procedure outputs the set of MG derivations that were reconstructed from the recovered (satisfiable) model interpretations – each (output) derivation accords with the (input) interface conditions, I, and can be generated from the (input) lexicon, $\mathcal{L}$.

Finally, we illustrate the parsing procedure with a worked out example. Consider the procedure taking as input the lexicon in Table 1 and the interface conditions (for the sentence *"What has the man eaten?"*) listed in entry I_1 of Table 2: after constructing the SMT model and constraining it with the input lexicon and interface conditions (detailed

[14] See (Harkema, 2001; Niyogi and Berwick, 2005; Stanojević, 2016; Torr et al., 2019) for earlier MG (chart) parsers.

[15] This further constrains the SMT model so that the solver cannot yield a model interpretation that encodes an MG derivation that the parser has already identified.

in §5), the procedure invokes the SMT-solver's model-checking (i.e. decision) routine to obtain the satisfiable model-interpretation presented in Table 3 (see also Appendix-Table 4); the procedure then recovers the output derivation shown in Fig. 1, which accords with I_1, from the satisfiable model-interpretation.

5 Specification of the SMT Model

This section details the SMT models of the MG derivation and MG lexicon - these models make up the heart of the parser introduced in this study.[16] These models consist of: (i) uninterpreted (i.e. free) finite sorts that represent model-objects such as words, syntactic features, categories, nodes in a derivation tree, etc; (ii) uninterpreted (free) functions that establish relationships between model-objects by mapping members of one or more sorts to another sort; (iii) model axioms – i.e. SMT-formulae – that constrain the valuation an SMT-solver may assign to each uninterpreted function.[17] (See Fig. 2 for a summary of the sorts and functions that make up the model.) Crucially, since the model of the parser has finite size (being bounded by the input parameter, p), we can explicitly quantify all of the SMT formulae in the model, *thereby yielding a decidable model that is solvable in finite time.*

We turn first to the **SMT model of the lexicon**. When constructing this model, the parsing procedure scans the input lexicon and instantiates several finite sorts: Σ, that models the set of PFs; $\mathbb{F}$, that models the set of feature-labels (e.g. $\{x, y, p, q\}$); and the *lexicon node sort*, Ω, that models the syntactic features appearing in the input lexicon.[18] The lexicon node sort is organized into disjoint subsets referred to as ***lexicon node sequences***, with each subset corresponding to one of the distinct lexical feature sequences appearing in the input lexicon.[19] Among the uninterpreted functions in the lexicon model, one plays an especially critical role: the successor function, ψ, which maps $a \in \Omega$ to $b \in \Omega$, where a corresponds to a node within a lexicon node sequence, and b corresponds to the subsequent node in that same lexicon node sequence;[20] the valuation of ψ is hard-coded by the parsing algorithm after Ω has been divided into lexicon node sequences.[21] The binary (uninterpreted) predicate, Δ_Ω, associates each lexical feature sequence with one or more (overt or covert) PFs, and these associations are hard-coded by the parsing procedure.[22] (E.g. Fig. 3 shows a lexicon node sequence and the lexical feature sequence it models.)

Next we turn to the **SMT model of the derivation**, which is composed of a finite sort, $\mathbb{N}$, that models the nodes in the derivation. The derivation takes the form of a *multi-dominance tree*[23] that is formed by augmenting the derivation tree with additional edges corresponding to the movement of phrases via internal merge (see Fig. 1). Members of $\mathbb{N}$ are sub-divided into ***derivation node sequences***, with each sequence corresponding to the projection of a lexical head within the derivation;[24] an important exception to this is a single member of $\mathbb{N}$, $\perp$, that serves as a *null-value* target for uninterpreted functions. The model's uninterpretable functions include:

(a) A unary function, p, that maps each node in a derivation node sequence to its successor node (in that sequence).

(b) A unary function, h, that maps each $x \in \mathbb{N}$ to the head (i.e. beginning) of the derivation node sequence to which x belongs; a derivation node $x \in \mathbb{N}$ is a head if and only if $h(h(x)) = h(x)$.

(c) A binary function, $\mathcal{M}$, that models *Merge*: given $x, y \in \mathbb{N}$, $\mathcal{M}(x, y)$ is the product of

[16] A complete, formal definition of these SMT models, including all model axioms, may be found in Ch. 2 of (Indurkhya, 2021a); see Appendix A for notes on reproducibility.

[17] All model axioms are written using propositional logic extended with (quantifier-free) theories of: (i) uninterpreted functions, (ii) Pseudo-Boolean constraints, and (iii) arithmetic.

[18] N.b. the sorts modeling the (fixed) sets of syntactic categories (e.g. N or V) and feature-types (e.g. $+$ or $=$) are pre-defined and do not depend on the input lexicon.

[19] E.g. the input lexicon in Table 1 has 29 distinct PFs, 4 distinct feature-labels, and 18 distinct lexical feature sequence, with each sequence having at most 3 features; therefore, the cardinality of the instantiated sorts Σ, $\mathbb{F}$ and Ω is 29, 4 and $18\times3 = 54$ (respectively).

[20] If a lexicon node $x \in \Omega$ corresponds to the terminal node in a lexicon node sequence, then $\psi(x) = x$.

[21] E.g. if, as in Fig. 3, $L_9, L_{14}, L_0, L_5 \in \Omega$ forms a lexicon node sequence that models the lexical feature sequence for entry 3 in Table 1, $[the :: =y, \sim y, -q]$, then the following constraint would be added to the SMT model of the lexicon: $(\psi(L_9){=}L_{14}) \wedge (\psi(L_{14}){=}L_0) \wedge (\psi(L_0){=}L_5)$.

[22] Encoding the SMT model of the lexicon with a representation that factors apart PFs and lexical feature sequences reduces the size of the model because lexical feature sequences are not duplicated, which in turn improves the performance of the SMT-solver. (E.g. in Table 1, the PFs for entries 24 and 28 will both map to the same lexicon node sequence.)

[23] Multi-dominance and derived trees are closely related (Kobele et al., 2007; Morawietz, 2008; Graf, 2013). Appendix-C details, and Appendix-Fig. 6 shows, how the *derivation node sequences* are organized so as to form a multi-dominance tree.

[24] *Derivation node sequences* are inspired by the closely related notion of "slices" (of a derivation tree) employed in Graf (2013). See Appendix-Fig. 6 for an illustration of how $\mathbb{N}$ is organized into derivation node sequences.

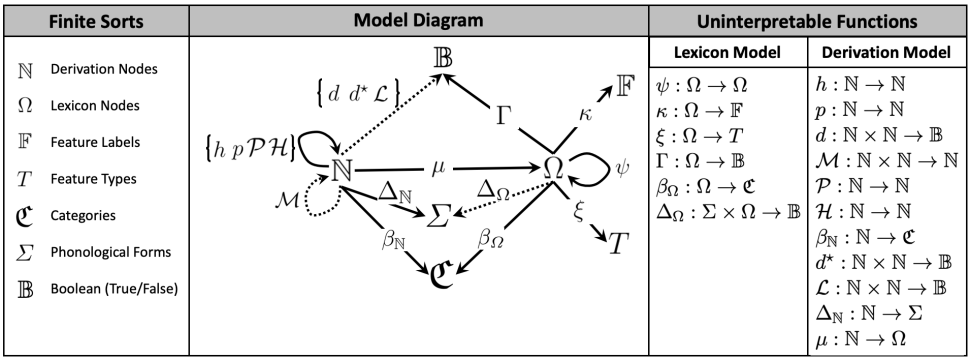

Figure 2: Arrangement of the uninterpreted functions and (finite) sorts that make up, and connect together, the SMT model of a derivation and the SMT model of a lexicon.

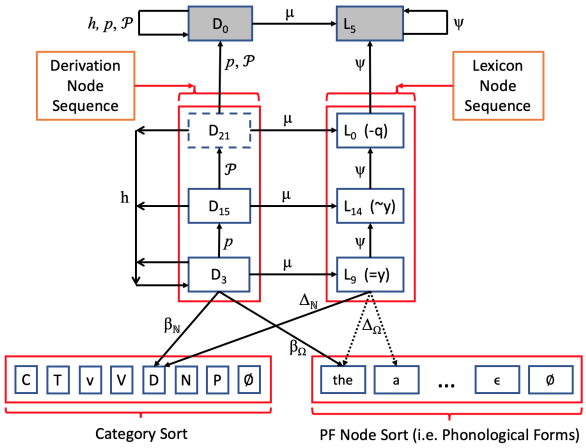

Figure 3: Model diagram showing how uninterpreted functions form commutative diagrams that connect the SMT model of the derivation to the SMT model of the lexicon – here they connect one of the *derivation node sequences* (from Fig. 1) to one of the *lexicon node sequences* (for entries 3 & 36 in Table 1). N.b. the lexicon node sequence maps to two PFs, and the derivation node sequence corresponds to one of those two PFs.

merging x with y.[25]

(d) A unary function, $\mathcal{P}$, that models the movement of phrases by mapping a node in the derivation tree to the location it is raised to.

(e) A unary function, $\mathcal{H}$, that models head-movement by mapping a lexical head to the lexical head that it incorporates with.

(f) Two binary predicates, d and $d^\star$, that encode the dominance relations making up the derivation (a multi-dominance tree), with d encoding dominance as imposed by p, and $d^\star$ encoding the dominance relations in the *derived tree* – i.e. the tree produced after all syntactic movement is completed (see Appendix-C for details).

(g) A unary function, $\beta_\mathbb{N}$, that associates each term in the derivation with a category (in $\mathfrak{C}$).

(h) A binary function, $\mathcal{L}$, encoding (linear) precedence (in accordance with the *derived tree*).

We restrict (satisfiable) interpretations of the SMT model by constraining it with additional axioms that encode various principles of minimalist syntax,[26] including axioms requiring:

(a) $\forall x, y \in \mathbb{N}, \mathcal{M}(x, y) = \mathcal{M}(y, x)$ (symmetry).

(b) no self-merging: $\forall x \in \mathbb{N}, \mathcal{M}(x, x) = \bot$.

(c) no term is the target of multiple merges: $\forall x, y, z \in \mathbb{N}, z \neq y \to \mathcal{M}(x, y) \neq \mathcal{M}(x, z)$.

(d) every non-lexical (i.e. non-leaf) node in the derivation tree is in the range of $\mathcal{M}$.

(e) $\forall x \in \mathbb{N}, h(\mathcal{P}(x)) = h(x)$.

(f) $\forall x, y \in \mathbb{N}$, if x and y are lexical heads related by head-movement (i.e. $(h(x) = x) \wedge (h(y) = y) \wedge (\mathcal{H}(x) = y)$), then the maximal projection of x is merged with y (via EM) - i.e. $\exists z \in \mathbb{N}$ s.t. $(h(z) = x) \wedge d(z, x) \wedge (h(\mathcal{M}(\mathcal{H}(x), z)) = y)$.

(g) the root node of the derivation tree is a (maximal) projection of a *complementizer* head (C), and the functional heads in a clause are organized as an *extended projection* of the form $C{\leftarrow}T{\leftarrow}v{\leftarrow}V$ (Adger and Svenonius, 2011).

(h) if a phrase, $x \in \mathbb{N}$, undergoes IM with a (lower) phrase, $y \in \mathbb{N}$, so that $\mathcal{P}(y)$ is the sister of x (i.e. $\mathcal{M}(\mathcal{P}(x), y) \neq \bot$), then $\mathcal{M}(x, \mathcal{P}(y)) = p(x)$ and $h(\mathcal{M}(x, \mathcal{P}(y))) = h(x) \neq h(y)$.

Notably, the expressive power of SMT, particularly the composition of uninterpretable functions, allows the model to consist of a few dozen axioms, which we found to be manageable to reason about.

[25] If x and y are not externally merged, then $\mathcal{M}(x, y) = \bot$; this illustrates one of the ways in which $\bot$ is utilized.

[26] E.g. the *Theory of Bare-Phrase Structure* (Chomsky, 1995), the *Inclusivess Principle* (Chomsky, 2001), the *No Tampering Condition* (Chomsky, 2005, 2013), the *Projection Principle* (Chomsky, 1986) and the *Principle of Economy of Derivation* (Collins, 2001).

Next the parsing procedure translates each of the (input) interface conditions (ICs) into SMT-formulae that constrain the SMT model of the derivation. LF ICs stipulating (subject-predicate) agreement and the assignment of θ-roles (i.e. semantic roles) to arguments are translated into model constraints (i.e. SMT-formulae) that require specific *local hierarchical relations* be established by *Merge*,[27] and the sentence type (i.e. declarative vs. interrogative) is translated into model constraints that dictate which type of complementizer, $C_{ques.}$ or $C_{decl.}$, heads the sentence. PF ICs are translated into constraints that require the *Subject-Verb-Object* (SVO) ordering of the derived tree, in which all phrasal-movement and head-movement has taken place, match the linear order of words in the input sentence.[28] Notably, the SMT-formulae encoding LF constraints are entirely separate from the SMT-formulae encoding PF constraints.

Finally, the SMT models of the lexicon and the derivation are connected by an uninterpreted function, μ, that maps each derivation node sequence to a lexicon node sequence, subject to the constraints: (i) $\mu \circ p = \psi \circ \mu$, which lines up each projection in the derivation with a lexical feature sequences (for a lexical entry) in the lexicon; (ii) $\beta_\Omega \circ \mu = \beta_\mathbb{N}$, which ensures that each lexical head in a derivation has the same category as the lexical entry it originates from. (Fig. 3 depicts these constraints and others as commutative diagrams.) There are also model-axioms that further restrict μ by requiring that pairs of nodes merged via EM or IM map to selectional or licensing features (respectively).

6 Parsing with Partially Specified Inputs

We validated the parsing procedure, and in particular the SMT-models it constructs, by using it to parse each pair of interface conditions in Table 2 using the lexicon in Table 1. Notably, this lexicon was designed so that, for each (LF, PF) pairing of interface conditions in Table 2, the lexicon can yield a derivation that satisfies the (input) interface conditions (ICs) and that matches the derivation prescribed by contemporary theories of minimalist syntax[29] – among these are derivations (in both active and passive voice) for declaratives, polar-interrogatives, *wh*-questions, relative clauses, and embedded sentences. Moreover, the (prescribed) derivations involve covert complementizers (C), tense-markers (T), and light-verbs (v), as well as various forms of movement including: *wh*-raising, subject-raising, T-to-C head-movement, and V-to-v head-movement (in *VP*-shells). The validation process succeeded, demonstrating that the parser, using the lexicon in Table 1, can yield (and internally model) the prescribed derivation for each entry in Table 2. E.g. see Fig. 7 & 8 for derivations, output by the parser, with an embedded sentence (for I_5) and a relative clause (for I_7), respectively.

We also measured, for each IC in Table 2, the

I_i	Interface Conditions	
I_1	PF:	what has the man/N eaten/V?
	LF:	$\theta_{\text{eaten}}[s\text{: the man}, o\text{: what}]$, $\mathcal{A}_{\text{has}}[s\text{: the man}]$
I_2	PF:	was she/N given/V money/N?
	LF:	$\theta_{\text{given}}[o\text{: money}, i\text{: she}]$, $\mathcal{A}_{\text{was}}[s\text{: she}]$
I_3	PF:	who will tell/V her/N that he/N has resigned/V?
	LF:	$\theta_{\text{tell}}[s\text{: who}, o\text{: that he has resigned}, i\text{: her}]$, $\mathcal{A}_{\text{will}}[s\text{: who}]$, $\theta_{\text{resigned}}[s\text{: he}]$, $\mathcal{A}_{\text{has}}[s\text{: he}]$
I_4	PF:	she/N has known/V everyone/N who was loved/V.
	LF:	$\theta_{\text{known}}[s\text{: she}, o\text{: everyone who was loved}]$, $\mathcal{A}_{\text{has}}[s\text{: she}]$, $\theta_{\text{loved}}[o\text{: everyone}]$, $\mathcal{A}_{\text{was}}[s\text{: everyone}]$
I_5	PF:	she/N knows/V that john/N has given/V money/N.
	LF:	$\theta_{\text{knows}}[s\text{: she}, o\text{: that john has given money}]$, $\theta_{\text{given}}[s\text{: john}, o\text{: money}]$, $\mathcal{A}_{\text{has}}[s\text{: john}]$
I_6	PF:	john/N has given/V money/N that was stolen/V.
	LF:	$\theta_{\text{given}}[s\text{: john}, o\text{: money that was stolen}]$, $\mathcal{A}_{\text{has}}[s\text{: john}]$, $\theta_{\text{stolen}}[o\text{: money}]$, $\mathcal{A}_{\text{was}}[s\text{: money}]$
I_7	PF:	john/N fears/V everyone/N who knows/V her/N.
	LF:	$\theta_{\text{fears}}[s\text{: john}, o\text{: everyone who knows her}]$, $\theta_{\text{knows}}[s\text{: everyone}, o\text{: her}]$
I_8	PF:	john/N fears/V that money/N was stolen/V.
	LF:	$\theta_{\text{fears}}[s\text{: john}, o\text{: that money was stolen}]$, $\theta_{\text{stolen}}[o\text{: money}]$, $\mathcal{A}_{\text{was}}[s\text{: money}]$

Table 2: Corpus of Paired (LF and PF) Interface Conditions (ICs). PF ICs provide surface order data, and some words are associated with a specified category (denoted by a slash followed by the category). LF ICs include relations for agreement ($\mathcal{A}$), predicate-argument structure (θ), and sentence-type (either declarative or interrogative as denoted by end-punctuation). N.b. LF ICs only encode hierarchical/structural relations – i.e. the values filling the slots consist of *sets* of tokens, not sequences of tokens. A predicate associates with one or more arguments: "s:" denotes an external argument, and "o:" and "i:" denote an internal argument serving as a direct or indirect object (respectively). Entries with an embedded clause – e.g. I_3 & I_8 – can have (separate) LF ICs stipulated for each clause.

[27] Specifically, per the *Uniformity of θ-Assignment Hypothesis* (Baker, 1988; Adger, 2003), internal (object or oblique) arguments are assigned a θ-role by establishing a local relationship (via EM) with the projection of a lexical verb, while external arguments are assigned a θ-role (e.g. AGENT) by establishing a local relationship with the light-verb within a VP-shell structure. Likewise, *subject-predicate agreement* requires a local relationship (established via IM) between a raised subject and the tense marker it agrees with.

[28] Following (Kayne, 1994), SVO ordering of the *derived tree* is obtained by requiring that *specifiers* precede their *head*, and that *heads* precede their *complement*.

[29] See (Adger, 2003; Hornstein et al., 2005; Hornstein and Pietroski, 2009; Collins and Stabler, 2016; Radford, 2016).

Node	$\beta_{\mathbb{N}}$	h	p	$\mathcal{P}$	$\mathcal{H}$	μ	$(\psi \circ \mu)$	$\Delta_{\mathbb{N}}$
D_0		D_0	D_0	D_0	D_0	L_5	L_5	
D_1	D	D_1	D_{12}	D_{19}	D_0	L_{37}	L_3	what
D_2	T	D_2	D_{14}	D_0	D_6	L_{32}	L_{36}	has
D_3	D	D_3	D_{15}	D_0	D_0	L_9	L_{14}	the
D_4	N	D_4	D_{15}	D_0	D_0	L_8	L_5	man
D_5	V	D_5	D_{12}	D_0	D_7	L_6	L_{33}	eaten
D_6	$C_{ques.}$	D_6	D_9	D_0	D_0	L_{23}	L_7	ϵ
D_7	v	D_7	D_{17}	D_0	D_0	L_{17}	L_4	ϵ
D_8		D_0	D_0	D_0	D_0	L_5	L_5	
D_9	C_{ques}	D_6	D_{22}	D_0	D_0	L_7	L_{27}	
D_{10}		D_0	D_0	D_0	D_0	L_5	L_5	
D_{11}		D_0	D_0	D_0	D_0	L_5	L_5	
D_{12}	V	D_5	D_{17}	D_0	D_0	L_{33}	L_5	
D_{13}	T	D_2	D_9	D_0	D_0	L_{24}	L_5	
D_{14}	T	D_2	D_{13}	D_0	D_0	L_{36}	L_{24}	
D_{15}	D	D_3	D_{18}	D_{21}	D_0	L_{14}	L_0	
D_{16}		D_0	D_0	D_0	D_0	L_5	L_5	
D_{17}	v	D_7	D_{18}	D_0	D_0	L_4	L_{35}	
D_{18}	v	D_7	D_{14}	D_0	D_0	L_{35}	L_5	
D_{19}	D	D_1	D_{22}	D_0	D_0	L_3	L_5	
D_{20}		D_0	D_0	D_0	D_0	L_5	L_5	
D_{21}	D	D_3	D_{13}	D_0	D_0	L_0	L_5	
D_{22}	$C_{ques.}$	D_6	D_0	D_0	D_0	L_{27}	L_5	

Table 3: Model interpretation for the derivation in Fig. 1. Each D_i is a member of the *derivation node sort*, $\mathbb{N}$. Valuations, recovered from the model interpretation, are listed for several of the uninterpreted functions (e.g. $h(D_{15})=D_3$ and $p(D_9)=D_{22}$) that make up: (i) the derivation model – i.e. h (head), p (parent), $\mathcal{P}$ (phrasal movement), $\mathcal{H}$ (head movement), $\Delta_{\mathbb{N}}$, and $\beta_{\mathbb{N}}$; (ii) the lexicon model – i.e. ψ (successor) and μ (bus). Not all members of $\mathbb{N}$ are used in the derivation (e.g. D_{11}); the bottom nodes, $D_0 \in \mathbb{N}$ and $L_0 \in \Omega$, serve as target nodes reserved for uninterpreted functions to map unused D_i to – e.g. $h(D_{11})=p(D_{11})=D_0$, and $\mu(D_{11})=L_0$.

runtime of the parser - i.e. the time the Z3 SMT-solver takes to check (i.e. solve) the constructed SMT model.[30] We found that I_1 and I_2 each took less than 12 seconds to parse, I_3-I_6 each took between 3 and 6 minutes to parse, and I_7 and I_8 took 31 and 41 minutes to parse (respectively). These differences in runtime are not unexpected when we observe that: (i) I_1 and I_2 have fewer tokens and no embedding structure (as compared to I_3-I_8); (ii) I_7 and I_8 require more instances of head-movement, empty categories and phrasal movement, so that the checked model is (substantively) larger than those of I_1-I_6. Moreover, we found in practice that there is a tradeoff between: (i) writing succinct, comprehensible model-axioms that make extensive use of compositions of uninterpretable functions, and (ii) the runtime of the Z3 SMT-solver. We believe navigating this tradeoff is an important avenue of future work for this parser, and that it is worth exploring the use of other higher-order theories supported by Z3, such as the theory of algebraic datatypes (Bjørner and Nachmanson, 2020), for modeling minimalist derivations and lexicons.

We next applied the parser to inputs in which either the LF or PF interface conditions are specified (but not both). We did this for each entry in Table 2, and present the analysis for I_1 below.

If the input is limited to the PF ICs in I_1, the parser can output a derivation (see Fig. 4) in which *"the man"* is the internal argument (as it merges with *"eaten"*) and *"what"* is the external argument (as it merges with the light-verb, v). This *alternative* derivation is possible because the external and internal arguments are selected using the same feature, $=y$, and swapping where the two arguments merge into the VP-shell structure compels the axiom encoding the *Uniformity of θ-Assignment Hypothesis* to assign semantic roles (to the arguments) that yield an incorrect reading of *"What has the man eaten?"* One solution is to refine the (selection) labels of nominal phrases (NP) to encode θ-roles; however, the model must be updated to propagate NP-labels to determiners (and complementizers and relative pronouns), or else the lexicon will grow untenably by multiplying out the determiners for each distinct selection label.

Conversely, if the input is instead limited to the LF ICs in I_1, then the parser can output a derivation (see Fig. 5) where the auxiliary verb *"has"* is not raised because T-to-C head-movement is compelled by PF ICs (and not by LF ICs); consequently, the surfaced form, *"What the man has eaten?"*, is ungrammatical. One solution is to add axioms that model Economy Conditions (Collins, 2001), so that T-to-C head-movement may be omitted if doing so leaves the surfaced form unchanged.

7 Conclusion

We have introduced an MG parser that is a computational model of HLF and is grounded in an SMT-model encoding a novel axiomatization of minimalist syntax. The parser uses the Z3 SMT-solver, an automatic theorem prover, to answer the question: *can the input lexicon yield a derivation that satisfies the input LF and PF interface conditions?* In this way, parsing is translated into an (SMT) decision problem, with model solutions corresponding to the derivations output by the parser.

We demonstrated that the parser, implemented within the *Parsing as Deduction* framework, can operate on partially specified interface conditions.[31]

[30] See Table 5 in the appendix for detailed results.

[31] More generally, we note that the flexibility of the parser's

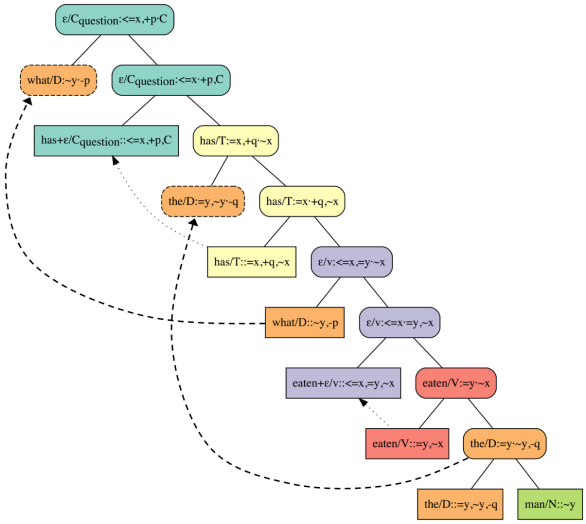

Figure 4: A derivation yielded by the parser, using the lexicon in Table 1, when only the PF interface conditions in entry I_1 (in Table 2) were input to the parser. In contrast with the (prescribed) derivation shown in Fig. 1, this derivation has the originating locations of the two arguments of the (lexical) verb *"eaten"* swapped; hence, although this derivation will be (correctly) externalized as *"What has the man eaten?"*, the derivation encodes an (incorrect) semantic interpretation in which the predicate *"eaten"* takes *"the man"* as its internal (object) argument, and *"what"* as its external (subject) argument (akin to the expression *"What has eaten the man?"*).

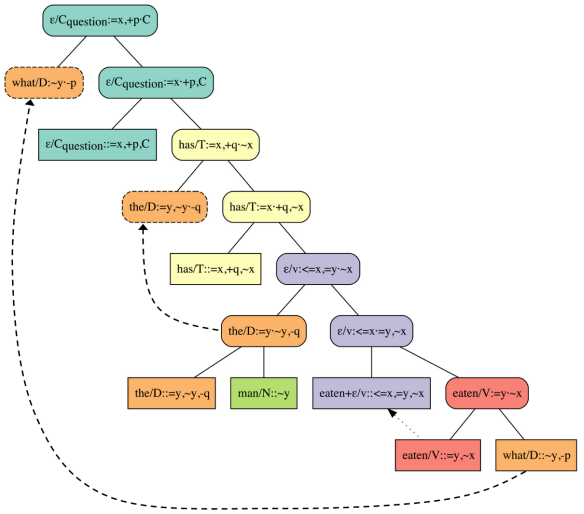

Figure 5: A derivation yielded by the parser, using the lexicon in Table 1, when only the LF interface conditions in entry I_1 (in Table 2) were input to the parser. In contrast with the (prescribed) derivation shown in Fig. 1, this derivation does not raise the auxiliary verb, *"has"*, via *T-to-C* head-movement; consequently, although this derivation accords with the LF interface conditions stipulated in I_1 (as it uses entry 36 in Table 1, which codes for an interrogative), it is externalized (i.e. surfaced) as the (un-grammatical) expression *"What the man has eaten?"*

This flexibility of the parser can be leveraged to observe when: (i) output derivations do not accord with the prescriptions of modern theories of minimalist syntax – inspecting these derivations can yield clues about how interface conditions and linguistic constraints cooperate to rule out derivations prohibited by the theory; (ii) the parser fails to output any derivation despite the theory prescribing a licit derivation – then the SMT-solver can identify the minimal subset of model-axioms that are mutually incompatible (Lynce and Silva, 2004; Guthmann et al., 2016), thus identifying conflicts between the axioms of minimalist syntax and the constraints derived from the interface conditions.

Finally, a key advantage of this parser is that it enables a division of labor: the SMT-solver is tasked with carrying out the logical deductions needed to find a model solution, leaving the linguist free to: (i) extend the parser, with the modular design of the SMT-model enabling related sets of axioms to be modified without impacting the remainder of the model;[32] (ii) investigate how principles of syntax cooperate to constrain the space of derivations, and identify redundant principles that may be dropped to yield a simpler theory of syntax.

Acknowledgements

I would like to thank three anonymous reviewers, as well as Robert C. Berwick, Sandiway Fong and Sanjoy Mitter for their comments and feedback.

design enables it to operate on partially specified inputs, with the SMT-solver in effect solving for the unspecified inputs (in addition to the derivation itself). E.g. if we specify the LF and PF interface conditions, but not the lexicon, then the parser will constrain the SMT model of the derivation using the interface conditions, but will not constrain the SMT model of the lexicon since no input lexicon was specified – then when the SMT-solver obtains a satisfiable interpretation of the SMT model of the parser, we can (automatically) recover from the interpretation of the lexicon model an MG lexicon that yields a derivation that satisfies the specified interface conditions. Moreover, if we augment the parser by connecting multiple SMT models of derivations, each constrained by a different pairing of interface conditions, to a single SMT model of a lexicon, then the composite SMT model can be used to infer an MG that can, for each pair of interface conditions, yield a derivation that satisfies that pairing – notably, this approach aligns with earlier work that used logic grammars to infer a lexicon (Rayner et al., 1988). See (Indurkhya, 2020) and (Indurkhya, 2022) for detailed discussions of how augmenting the parser in this manner can yield instantaneous and incremental (respectively) computational models of language acquisition.

[32] E.g. to support *head-final* languages (e.g. French or Japanese), the model-axioms encoding *SOV* ordering can be replaced (with axioms for *SOV* ordering) without altering model-axioms unrelated to the PF-interface (see Appendix D).

References

David Adger. 2003. *Core syntax: A minimalist approach*, volume 33. Oxford University Press Oxford.

David Adger and Peter Svenonius. 2011. Features in minimalist syntax. *The Oxford handbook of linguistic minimalism*, pages 27–51.

Krzysztof R Apt. 1990. Logic programming. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 1990:493–574.

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

Nikolaj Bjørner. 2011. Engineering theories with z3. In *Asian Symposium on Programming Languages and Systems*, pages 4–16. Springer.

Nikolaj Bjørner and Lev Nachmanson. 2020. Navigating the universe of z3 theory solvers. In *Brazilian Symposium on Formal Methods*, pages 8–24. Springer.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Studies in generative grammar. Foris.

Noam Chomsky. 1986. *Knowledge of language: Its nature, origin, and use*. Greenwood Publishing Group.

Noam Chomsky. 1995. *The Minimalist Program*. Volume 28 of Current studies in linguistics series. MIT Press.

Noam Chomsky. 2001. Derivation by phase. In Michael Kenstowicz, editor, *Ken Hale: A life in language*, pages 1–52. MIT Press.

Noam Chomsky. 2005. Three factors in language design. *Linguistic Inquiry*, 36(1):1–22.

Noam Chomsky. 2013. Problems of projection. *Lingua*, 130:33–49.

Chris Collins. 2001. Economy conditions in syntax. *The Handbook of Contemporary Syntactic Theory*, pages 45–61.

Chris Collins and Edward Stabler. 2016. A formalization of minimalist syntax. *Syntax*, 19(1):43–78.

Leonardo De Moura and Nikolaj Bjørner. 2011. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77.

Ralph Debusmann, Denys Duchier, Marco Kuhlmann, and Stefan Thater. 2004. Tag parsing as model enumeration. In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 148–154.

Denys Duchier. 1999. Axiomatizing dependency parsing using set constraints. In *Sixth Meeting on the Mathematics of Language*, page 115–126.

Denys Duchier, Thi-Bich-Hanh Dao, Yannick Parmentier, and Willy Lesaint. 2010. Property grammar parsing seen as a constraint optimization problem. In *Formal Grammar*, pages 82–96. Springer.

Bruno Dutertre and Leonardo de Moura. 2006. A fast linear-arithmetic solver for DPLL(T). In *Computer Aided Verification*, pages 81–94, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mathieu Estratat and Laurent Henocque. 2004. Parsing languages with a configurator. In *European Conference on Artificial Intelligence*, volume 16, page 591.

Sandiway Fong. 1991. *Computational properties of principle-based grammatical theories*. Ph.D. thesis, Massachusetts Institute of Technology.

Thomas Graf. 2011. Closure properties of minimalist derivation tree languages. In *International Conference on Logical Aspects of Computational Linguistics*, pages 96–111. Springer.

Thomas Graf. 2013. *Local and transderivational constraints in syntax and semantics*. Ph.D. thesis, University of California at Los Angeles.

Jane B Grimshaw. 2005. *Words and structure*. CSLI Publications.

Ofer Guthmann, Ofer Strichman, and Anna Trostanetski. 2016. Minimal unsatisfiable core extraction for smt. In *2016 Formal Methods in Computer-Aided Design (FMCAD)*, pages 57–64.

Kenneth Hale and Samuel J. Keyser. 1993. On argument structure and the lexical expression of syntactic relations. In Kenneth Hale and Samuel J. Keyser, editors, *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*, pages 53–109. MIT Press.

Kenneth Hale and Samuel Jay Keyser. 2002. *Prolegomenon to a theory of argument structure*, volume 39 of *Linguistic Inquiry Monographs*. MIT Press.

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

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

Norbert Hornstein and Paul Pietroski. 2009. Basic operations: Minimal syntax-semantics. *Catalan Journal of Linguistics*, 8(1):113–139.

Sagar Indurkhya. 2020. Inferring Minimalist Grammars with an SMT-solver. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 457–460, New York, New York. Association for Computational Linguistics.

Sagar Indurkhya. 2021a. *Solving for syntax*. Ph.D. thesis, Massachusetts Institute of Technology.

Sagar Indurkhya. 2021b. Using collaborative filtering to model argument selection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 629–639, Held Online. INCOMA Ltd.

Sagar Indurkhya. 2022. Incremental acquisition of a Minimalist Grammar using an SMT-solver. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 212–216, online. Association for Computational Linguistics.

Sagar Indurkhya and Robert C. Berwick. 2021. Evaluating the cognitively-related productivity of a universal dependency parser. In *2021 IEEE 20th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pages 7–15.

Joxan Jaffar and J-L Lassez. 1987. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 111–119.

Joxan Jaffar and Michael J Maher. 1994. Constraint logic programming: A survey. *The Journal of Logic Programming*, 19:503–581.

Mark Johnson. 1989. Parsing as deduction: The use of knowledge of language. *Journal of Psycholinguistic Research*, 18(1):105–128.

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

Richard S Kayne. 1994. *The antisymmetry of syntax*. 25. MIT Press.

Gregory M Kobele. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *International Conference on Logical Aspects of Computational Linguistics*, pages 129–144. Springer.

Gregory M Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. *Model Theoretic Syntax at 10*, pages 71–80.

Alexander Koller and Joachim Niehren. 2002. Constraint Programming in Computational Linguistics. In Dave Barker-Plummer, David I. Beaver, Johan van Benthem, and Patrick Scotto di Luzio, editors, *Words, Proofs, and Dialog*, volume 141, pages 95–122. CSLI Press.

Yuliya Lierler and Peter Schüller. 2012. Parsing combinatory categorial grammar via planning in answer set programming. In *Correct Reasoning*, pages 436–453. Springer.

Inês Lynce and João Silva. 2004. On computing minimum unsatisfiable cores. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 305–310.

Jens Michaelis. 1998. Derivational minimalism is mildly context-sensitive. In *International Conference on Logical Aspects of Computational Linguistics*, volume 98, pages 179–198. Springer.

Jens Michaelis. 2001. Transforming linear context-free rewriting systems into minimalist grammars. *Logical Aspects of Computational Linguistics*, pages 228–244.

Jens Michaelis, Uwe Mönnich, and Frank Morawietz. 2000. Algebraic description of derivational minimalism. *Algebraic Methods in Language Processing*, 16.

Frank Morawietz. 2008. *Two-Step Approaches to Natural Language Formalism*, volume 64. Walter de Gruyter.

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer.

Leonardo de Moura and Nikolaj Bjørner. 2009. Satisfiability modulo theories: An appetizer. In *Brazilian Symposium on Formal Methods*, pages 23–36. Springer.

Robert Nieuwenhuis and Albert Oliveras. 2006. On sat modulo theories and optimization problems. In *International conference on theory and applications of satisfiability testing*, pages 156–169. Springer.

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. 2006. Solving sat and sat modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM (JACM)*, 53(6):937–977.

Sourabh Niyogi and Robert C Berwick. 2005. A minimalist implementation of Hale-Keyser incorporation theory. In *UG and External Systems: Language, Brain and Computation*, pages 269–288. John Benjamins Publishing.

Fernando C. N. Pereira and David H. D. Warren. 1983. Parsing as deduction. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, ACL '83, pages 137–144. Association for Computational Linguistics.

David Pesetsky and Esther Torrego. 2001. T-to-c movement: Causes and consequences. *Current Studies in Linguistics Series*, 36:355–426.

Andrew Radford. 2009. *An introduction to English sentence structure*. Cambridge University Press.

Andrew Radford. 2016. *Analysing English sentences: A minimalist approach (Second Edition)*. Cambridge University Press.

Silvio Ranise and Cesare Tinelli. 2006. Satisfiability modulo theories. *Trends and Controversies-IEEE Intelligent Systems Magazine*, 21(6):71–81.

Manny Rayner, Asa Hugosson, and Goran Hagert. 1988. Using a logic grammar to learn a lexicon. In *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*.

James Rogers and Rachel Nordlinger. 1998. *A descriptive approach to language-theoretic complexity*. MIT press.

Yves Schabes and Richard C Waters. 1993. Lexicalized context-fee grammars. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 121–129.

Peter Schüller. 2013. Flexible combinatory categorial grammar parsing using the cyk algorithm and answer set programming. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 499–511. Springer.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(1-2):3–36.

Edward Stabler. 1996. Derivational minimalism. In *International Conference on Logical Aspects of Computational Linguistics*, pages 68–95. Springer.

Edward P Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*, 5(3):611–633.

Edward P. Stabler and Edward L Keenan. 2003. Structural similarity within and among languages. *Theoretical Computer Science*, 293(2):345–363.

Miloš Stanojević. 2016. Minimalist grammar transition-based parsing. In *International Conference on Logical Aspects of Computational Linguistics*, pages 273–290. Springer.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*, pages 181–224.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.

John Torr, Milos Stanojevic, Mark Steedman, and Shay B Cohen. 2019. Wide-coverage neural a* parsing for minimalist grammars. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 2486–2505.

A Reproducibility

We ran the computer programs detailed in this study on a MacBook Pro (Retina, 15-inch, Late 2013) with a 2.3 GHz Intel Core i7 processor, and 16GB of 1600MHz DDR3 RAM. We used Python v3.7.9 and v4.8.6 of the Z3 SMT-solver. The complete program source code for the parser, including the (python) source code for the SMT models, is available at `https://github.com/indurks/mgsmt`.

B Minimalist Grammar

This section provides additional details about the Minimalist Grammar formalism used in the present study. Notably, MGs are mildly-context sensitive (Michaelis, 1998) and are sufficiently expressive for modeling natural language in so far as they can model the syntactic constraints that appear in contemporary syntax (e.g. they can produce structures encoding cross-serial dependencies) – specifically, the syntactic constraints underlying HLF can be modeled by Monadic Second Order (MSO) logic (Rogers and Nordlinger, 1998), and MSO-expressible constraints over an MG derivation tree can be encoded within an MG lexicon (Graf, 2013).[33]

We now turn to reviewing the algebraic formulation of MGs presented in Stabler and Keenan (2003) – we encourage the reader to consult Fig. 1 and Table 1 to ground this formal presentation. A minimalist grammar, G, is defined by a tuple, $(\Sigma, Sel, Lic, Lex, \mathbb{M})$, and we will now define each member of this tuple in turn. First, Σ is a finite, non-empty set of phonological forms – a phonological form is either *overt* (i.e. a pronounced word) or *covert* (i.e. unpronounced), and we let ϵ denote a covert phonological form. Next, Sel and Lic are defined as non-empty (disjoint) finite sets of feature labels for *selection* and *licensing* respectively.[34] We then define F, the set of syntactic features, as the union of:

(i) the singleton set containing the special feature C, which marks the end of the derivation process;

(ii) the set of selectional features, formed by prefixing members of Sel with $=$ or $\sim$ to indicate if the feature is a *selector* or a *selectee* (respectively); furthermore, a $<$ or $>$ prefixed before a selector prefix – i.e. "<=" or ">=" – indicates that the selector triggers left or right head-movement respectively.[35]

(iii) the set of licensing features, formed by prefixing members of Lic with $+$ or $-$ to indicate if the feature is a *licensor* or a *licensee* (respectively).

Turning to the lexicon, Lex, we first define the set of *chains* as $H = \Sigma^* \times Types \times F^*$, where the set $Types = \{::, :\}$ designates whether a chain is *lexical* or *derived* (from *lexical* chains) respectively.[36] We can then define Lex as a non-empty finite set of lexical chains. Finally, the set of expressions, $E = H^+$, may be recursively combined together via the binary structure building operation *Merge*, denoted by $\mathbb{M}$, to produce another expression. *Merge* has two disjoint subcases:

(i) *external merge* (EM), which models combination, requires that both arguments of merge are disjoint from one another;

(ii) *internal merge* (IM), which models displacement, requires that one of the arguments is a constituent of the other.

Both sub-cases of *Merge* are driven by feature-checking, with M determining whether two expressions may be paired together based on their features; note that the syntactic features are uninterpretable, and *Merge* deletes the pairs of features that check one another.

Let us now formally detail the subcases of M.

[33] Notably, MGs are sufficiently expressive for modeling syntactic derivations that are systematically related by structural transformations. E.g. a declarative is (structurally) related to its corresponding polar-interrogative by way of the rule for *aux-raising* (i.e. T-to-C movement as modeled in contemporary minimalist syntax) in which the top most (i.e. root) complementizer triggers head-movement of the (hierarchically) closest tense-marker – we would thus expect that the syntactic structure assigned (by an MG parser) to a declarative could be transformed into a polar-interrogative by replacing lexical item 25 with lexical item 9 (in Table 1), and would also expect that running an MG parser on the polar-interrogative would yield the same derivation as obtained by applying *aux-raising* to the derivation of the declarative. This capability of MGs and their parsers stands in contrast with state-of-the-art UD parsers that have difficulty acquiring and encoding knowledge of the *aux-raising* rule (Indurkhya and Berwick, 2021).

[34] The feature system used here is based on checking theory as detailed in Chomsky (1995).

[35] Instances of head-movement include: (i) the V-to-v head-movement utilized in the Hale-Keyser model of predicate-argument structure (Hale and Keyser, 1993, 2002); (ii) T-to-C head-movement (Pesetsky and Torrego, 2001) that is utilized in fronting an auxiliary verb (e.g. when forming a polar-interrogative from a declarative).

[36] *Lexical chains* serve to track the sequence of movement operations that the (maximal) projection (of a lexical head) may undergo in the course of a derivation; in particular, they track terms in the derivation that have not yet finished moving (and thus need to be accessible to the Internal Merge operation.

Let $s, t \in \Sigma^*$, $f \in Sel$, $g \in Lic$, $\gamma \in F^*$ and $\delta \in F^+$. Furthermore, let $\alpha_1, ..., \alpha_k \in H$ for $0 \leq k$, and let $\iota_1, ..., \iota_l \in H$ for $0 \leq l$. We then define EM as the union of the following three (disjoint) functions, $\{EM_1, EM_2, EM_3\}$, that involve feature *selection*:

$$\frac{[\text{s} :: = f, \gamma] \qquad [\text{t} \cdot \sim f], \iota_1 ... \iota_l}{[\text{st} : \gamma], \iota_1 ... \iota_l} \text{ EM}_1$$

$$\frac{[\text{s} : = f, \gamma], \alpha_1 ... \alpha_k \qquad [\text{t} \cdot \sim f], \iota_1 ... \iota_l}{[\text{ts} : \gamma], \alpha_1 ... \alpha_k, \iota_1 ... \iota_l} \text{ EM}_2$$

$$\frac{[\text{s} \cdot = f, \gamma], \alpha_1 ... \alpha_k \qquad [\text{t} \cdot \sim f, \delta], \iota_1 ... \iota_l}{[\text{s} : \gamma], \alpha_1 ... \alpha_k, [\text{t} : \delta], \iota_1 ... \iota_l} \text{ EM}_3$$

The separation of the phonological form and the syntactic features by the symbol $\cdot$ designates that the chain could either be *lexical* or *derived*. IM is defined as the union of the two disjoint functions, $\{IM_1, IM_2\}$, that employ feature licensing:

$$\frac{[\text{s} : + g, \gamma], \alpha_1 ... \alpha_{i-1}, [\text{t} : - g], \alpha_{i+1} ... \alpha_k}{[\text{ts} : \gamma], \alpha_1 ... \alpha_{i-1}, \alpha_{i+1} ... \alpha_k} \text{ IM}_1$$

$$\frac{[\text{s} : + g, \gamma], \alpha_1 ... \alpha_{i-1}, [\text{t} : - g, \delta], \alpha_{i+1} ... \alpha_k}{[\text{s} : \gamma], \alpha_1 ... \alpha_{i-1}, [\text{t} : \delta], \alpha_{i+1} ... \alpha_k} \text{ IM}_2$$

Furthermore, IM_1 and IM_2 are restricted by the *Shortest Move Constraint* (SMC): if a licensor, α, binds to a licensee, β, it must be the case that β is the only licensee to which α can bind. The SMC ensures that the licensor will always select the (hierarchically) nearest licensee, as at every step in the derivation, there can only be one possible licensee that can be licensed; this has the consequence of making IM deterministic (with respect to which licensee a licensor will license), so that a derivation can be determined entirely from knowledge of the order in which the various lexical heads (and projections thereof) are *externally* merged with one another.

Finally, we define a *derivation* as a sequence of expressions produced by recursively applying $\mathbb{M}$ to a group of chains; a derivation is deemed to be *complete* if there remains a single expression that has no chains and that has one feature, C (which serves to indicate the termination point of the derivation).[37]

[37] As defined here, an MG either can or cannot generate a given derivation. However, we can compute a relative likelihood for a given derivation to be generated by an MG by determining for each of the merge operations involving (constituent) selection (i.e. the *c-selection* that drives external merge), the degree to which the heads of the two merged projections tend to associate with one another – this pairwise associativity between phonological forms (corresponding to the two heads) can be computed by various methods, e.g. using a similarity metric to compute distance between the word embedding vectors for the two phonological forms, or using model-based collaborative filtering may be used to compute the associativity between predicates and arguments (Indurkhya, 2021b).

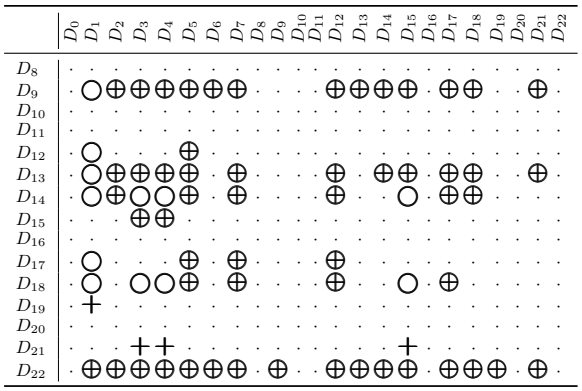

Table 4: Model interpretation of two binary uninterpreted functions, d and $d^\star$, for the derivation in Fig. 1. Given an entry at row D_i and column D_j: $\bigoplus$ indicates that the node D_i dominates the node D_j with respect to the derived tree but not the derivation tree; $\bigcirc$ indicates that D_i dominates D_j with respect to the derivation tree but not the derived tree; $\bigoplus$ indicates that D_i dominates D_j with respect to both the derivation tree and the derived tree; and $\cdot$ indicates that D_i does not dominate D_j (with respect to either the derivation tree or the derived tree). E.g. D_{18} dominates D_{15} with respect to the derivation tree but not the derived tree: notice in Table 3 that while $p(D_{15}) = D_{18}$, there is no $k \in [0, 22]$ such that $\mathcal{P}(D_k) = D_{18}$. Conversely, D_{21} dominates D_{15} with respect to the derived tree but not the derivation tree: notice in Table 3 that $\mathcal{P}(D_{15}) = D_{21}$, but there is no $k \in [0, 22]$ such that $p(D_k) = D_{21}$. The derivation's root node, D_{22}, dominates each of the other nodes in the derivation with respect to both the derivation tree and the derived tree. Finally, $D_1, D_2, \ldots, D_7$, which are leaf nodes (i.e. lexical heads) in the derivation, do not dominate any other nodes in the derivation, and for that reason rows $D_1 \ldots D_7$ are not shown as they would be entirely filled by $\cdot$.

Assuming a *Subject-Verb-Object* word-ordering, the surface form associated with a complete derivation may be read out by recursively applying (top-down) a Specifier-Head-Complement linearization of each projection.[38]

C Multi-dominance and Derived Trees

This section details how a *minimalist derivation* takes the form of a *multi-dominance tree* – i.e. the *(bare) phrase structures* that linguists are familiar

[38] In a projection of a lexical head, the complement is the first term the lexical head merges with, and the specifier is the subsequent term that the projection (of the head) merges with – e.g. in XBar-theoretic terms, given the two rules: $XP \rightarrow Spec, X'$, and $X' \rightarrow X, Comp$, the projection of the lexical head X will be linearized so that the surface ordering is $Spec, X, Comp$.

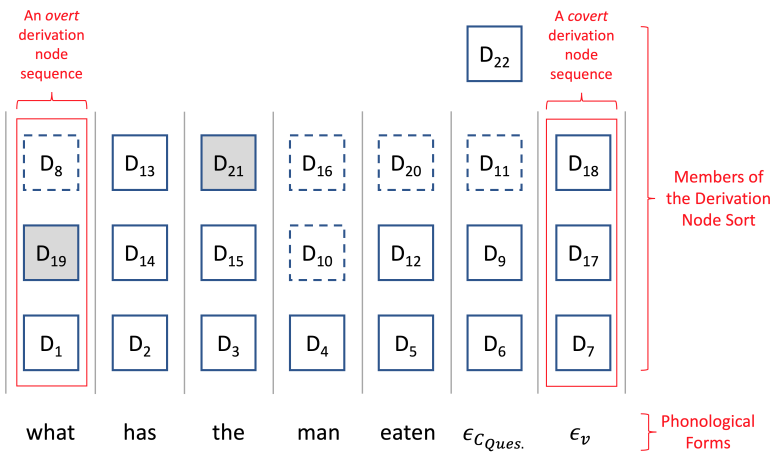

Figure 6: An illustration of how the members of the derivation node sort, $\mathbb{N}$, are arranged into *derivation node sequences*, with each sequence being associated with either an overt or covert phonological form. Each derivation node sequence is depicted as a column, with the first node in the sequence at the bottom and the last node in the sequence at the top. Note that the derivation node sequences shown here may be arranged so as to form the derivation (tree) shown in Fig. 1. Nodes that actively play a role in the derivation are depicted as white boxes, and active nodes that are in the same column have the same (lexical) head - e.g. the root node is D_{22}, and since D_{22} has the same head as D_9 and D_6, it is displayed here above the covert node-sequence associated with the (covert) phonological form $\epsilon_{C_{ques.}}$. (Note that the root node is not a member of any derivation node sequence, and is treated as a special case in the axioms.) Boxes with dashed boundaries correspond to inactive members of $\mathbb{N}$ that do not participate in the derivation (i.e. they do not appear in the derivation in Fig. 1). Boxes with solid boundaries are projections, whereas greyed out boxes are part of a lexical chain (i.e. the sequence of movement operations that a maximal projection may participate in). Importantly, the derivation node sequences together form an index over $\mathbb{N}$, and this index enables us to write model axioms that can explicitly reference the members of a *derivation node sequence* – i.e. the axioms that constrain uninterpreted functions operating over $\mathbb{N}$ can explicitly reference each individual step in the projection (and potential subsequent chain) of the lexical head associated with a given phonological form.

with.[39]

A multi-dominance tree is a super-position of the *derivation tree* – i.e. the tree made up of the external and internal merge operations that work together to combine a multi-set of lexical items drawn from the lexicon – and the *derived tree*, which is the tree that remains after a minimalist derivation has been generated and all movement operations have been applied. Each MG derivation tree is associated with a *multi-dominance tree*, which can be generated from the derivation tree by appending, for each occurrence of IM in the derivation tree, a node at the destination of the movement operation, and then establishing a dominance relation (via $d^\star$) between the destination node and the node at the source of movement.[40]

We observe that, for both the derivation and multi-dominance trees, each node is associated with a (lexical) *head*; then, since two nodes that are merged together cannot have the same head, we can identify which of two merged constituents projects by examining the head of the node that corresponds to the product of merge.[41]

- The *derivation tree* can be recovered from the multi-dominance tree by deleting each occurrence of movement (i.e. deleting the node at the raised location).
- The *derived tree* may be recovered from the multi-dominance tree by removing, for each node x in the multi-dominance tree that serves as a source of movement, the dominance relation (with respect to the derived tree) between

[39] Relatedly, see Pgs. 12-24 of Graf (2013) for a discussion of *"augmented derivation trees."*

[40] This is closely related to the two-step approach that involves first lifting information *implicitly* encoded within a derivation tree (i.e. the information encoded in the structure of the multi-dominance tree) so as to to make the information explicit, and then reconstructing the (derived) phrase structure tree that linguists are more familiar with. See Pgs. 35-50 of Graf (2013) for a discussion of the two-step approach of (i) lifting an MG derivation to its associated the multi-dominance tree and then (ii) reconstructing the "derived tree"; see also (Kobele et al., 2007). See Morawietz (2008) (Pgs. 131-182) for a review of the two-step approach as applied to multiple context-free grammars (MCFGs), and note that MGs may be translated into MCFGs (Michaelis et al., 2000).

[41] N.b. the derivation and multi-dominance trees do not explicitly encode (linear) precedence relations between the lexical heads entering into the derivation.

x and its parent – i.e.:

$$d^{\star}(p(x), x) = \text{False}$$

Importantly, the multi-dominance tree can be viewed as a super-position of the derivation tree and the derived tree, and it is the multi-dominance tree associated with an MG derivation that serves as the domain of discourse in the SMT model of the derivation. Hence, whenever the present study refers to a derivation tree or a derived tree, the reader should understand that they are components of a multi-dominance tree.

Each *lexical item* that appears in a derivation has a (bottom-up) trajectory through the associated multi-dominance tree:

(i) the lexical item, starting as a lexical head, is first projected zero or more times – this process is driven by either external merge via (c-)selection or internal merge via licensing;

(ii) the (maximal) projection of the lexical item is then either the terminal point of the derivation (marked by the presence of the special symbol C) or is selected by some other lexical head (this is driven by the presence of a selectee feature);

(iii) finally, the lexical item is raised, via internal merge, zero or more times to form a movement-chain, with each movement operation forming a link in the chain.

Importantly, there are two key points to take away from this observation:

(a) Each node in the multi-dominance tree associates with a lexical item in the derivation (i.e. the lexical item that is the head of that node) and the nodes associated with a lexical head may be arranged as a sequence in the order in which they appear in the multi-dominance tree (starting from the bottom); ***for this reason, we refer to such a sequence as a "derivation node sequence" and observe that the multi-dominance tree associated with an MG derivation is a structural arrangement of derivation node sequences*** (Stabler, 2013).

(b) Given the multi-dominance tree that is associated with an MG derivation, we can recover the multiset of lexical items from which the multi-domimance tree is derived (except for the labels of the syntactic features); this can be seen by observing that each node in a derivation node sequence is associated with exactly one type of syntactic feature – i.e. selector, selectee, licensor, licensee, or the special symbol C – and noting that the feature-type of a node can be determined by the position of that node within the multi-domimance tree, so that given a derivation node sequence associated with a lexical entry, the corresponding sequence of syntactic feature-types (present in that lexical entry) can be obtained the path that the derivation node sequence takes through the multi-dominance tree.

(See Fig. 6 for an illustration of the derivation node sequences that are assembled to form the derivation presented in Fig. 1.) ***Consequently, an SMT model of a minimalist derivation can be constructed by: (i) modeling the derivation node sequences that form the associated multi-dominance tree, and (ii) constraining the topology of the multi-domimance tree by using the model axioms to restrict how the derivation node sequences may be assembled together.***

IC	Trial 1	Trial 2	Trial 3	Median
I_1	11.7	10.5	13.9	11.7
I_2	3.2	3.3	4.0	3.3
I_3	323.8	208.9	346.0	323.8
I_4	267.1	296.2	281.1	281.1
I_5	222.6	225.5	178.5	222.6
I_6	261.8	312.0	261.4	261.8
I_7	1213.3	2065.6	1857.2	1857.2
I_8	2445.1	1851.7	3275.9	2445.1

Table 5: Runtime performance, measured in seconds, of the parser (i.e. the time Z3 takes to check the constructed SMT-model of the parser).

D Limitations

This section briefly comments on two limitations of the parser introduced in this study.

One limitation of the parser is that it has only been tested on (Modern Standard) English, which has Subject-Verb-Object (SVO) ordering; however, we believe that the parser can be readily adapted to languages with Subject-Object-Verb (SOV) ordering (e.g. French or Japanese) by replacing a small number of the constraints (derived from PF interface conditions) that encode SVO-ordering by applying *Specifier-Head-Complement* linearization to the derived tree: namely, these constraints for SVO-ordering could be replaced with constraints that enforce SOV-ordering based on applying *Specifier-*

Complement-Head linearization (see the relevant footnote in §7). Moreover, it would be interesting to investigate whether the SMT model of the parser could be augmented with a (boolean) variable that serves as a switch, controlling whether the constraints for SVO or SOV are used; notably, such a switch could either be hard-coded by the user (to enforce which ordering the parser should use), or left un-valued, in which case the parser could use either (SVO or SOV) ordering, so long as the surfaced word-sequence (yielded by the output derivation) aligns with the input word-sequence (so that the input PF interface conditions are satisfied).

Another limitation of the parser is that it is primarily focused on modeling syntax, and does not explicitly model morphological inflection. We believe that, in future work, this limitation could be overcome (in part) by: (i) augmenting the SMT model of the lexicon to store the *root* of each (overt) phonological form and encoding morphological attributes within the labels of the syntactic features; (ii) updating the constraints (i.e. SMT-formulae) derived from the PF interface conditions to inflect each root form when comparing it against the relevant surface form (i.e. the inflected word listed in the input PF interface conditions) - this inflection would be realized by the constraints inspecting the morphological attributes encoded in the feature label associated with that root form.

We believe that both of these (current) limitations point to productive avenues for further research involving extending the parser presented in this study.

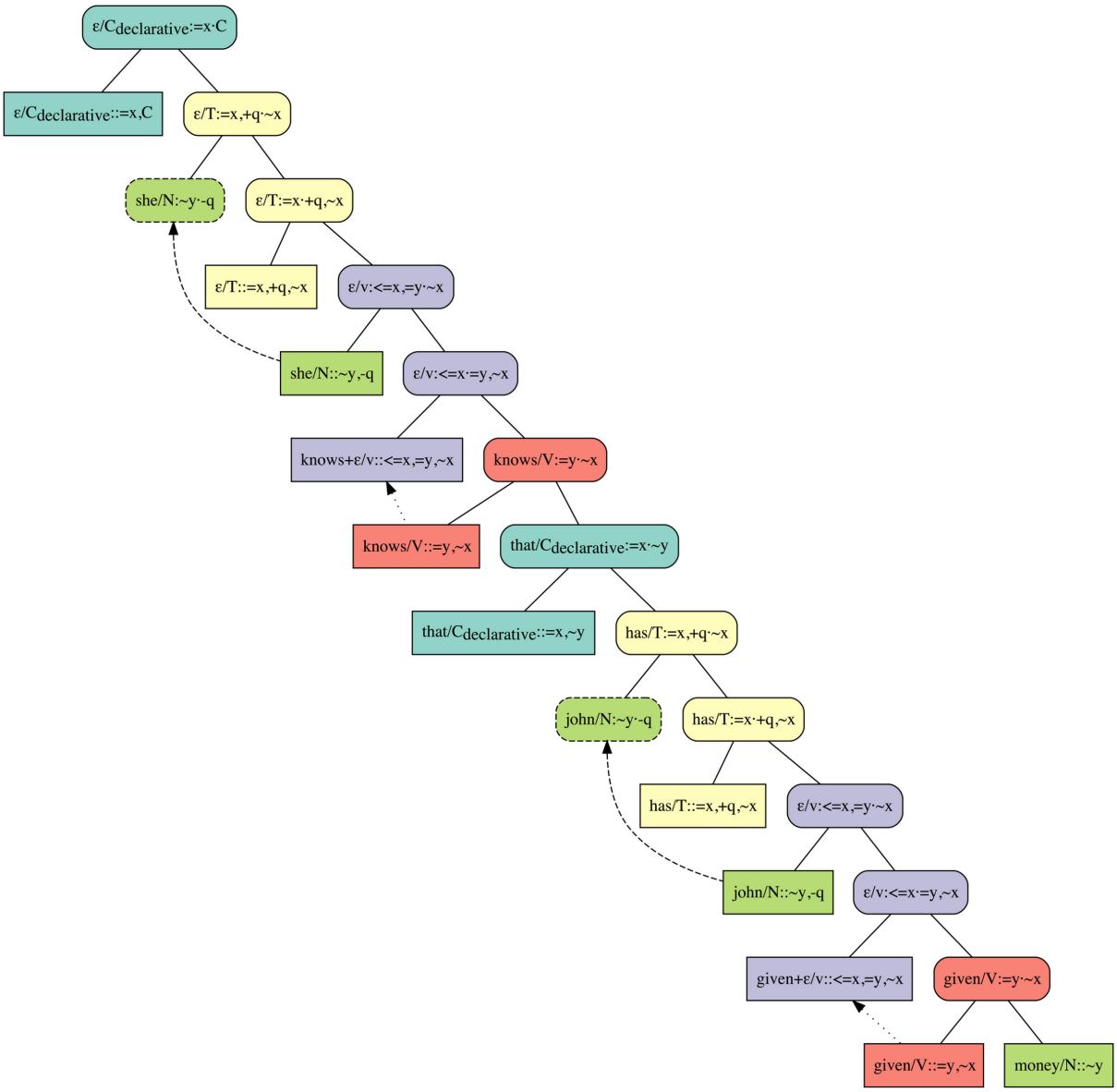

Figure 7: A derivation for the sentence: *"She knows that John has given money."* This derivation was output by the parser when it was applied to entry I_5 in Table 2, using the lexicon in Table 1, and matches the derivation prescribed by contemporary theories of minimalist syntax. This demonstrates the parser's capacity to model an input with an embedded sentence – i.e. *"John has given money"*.

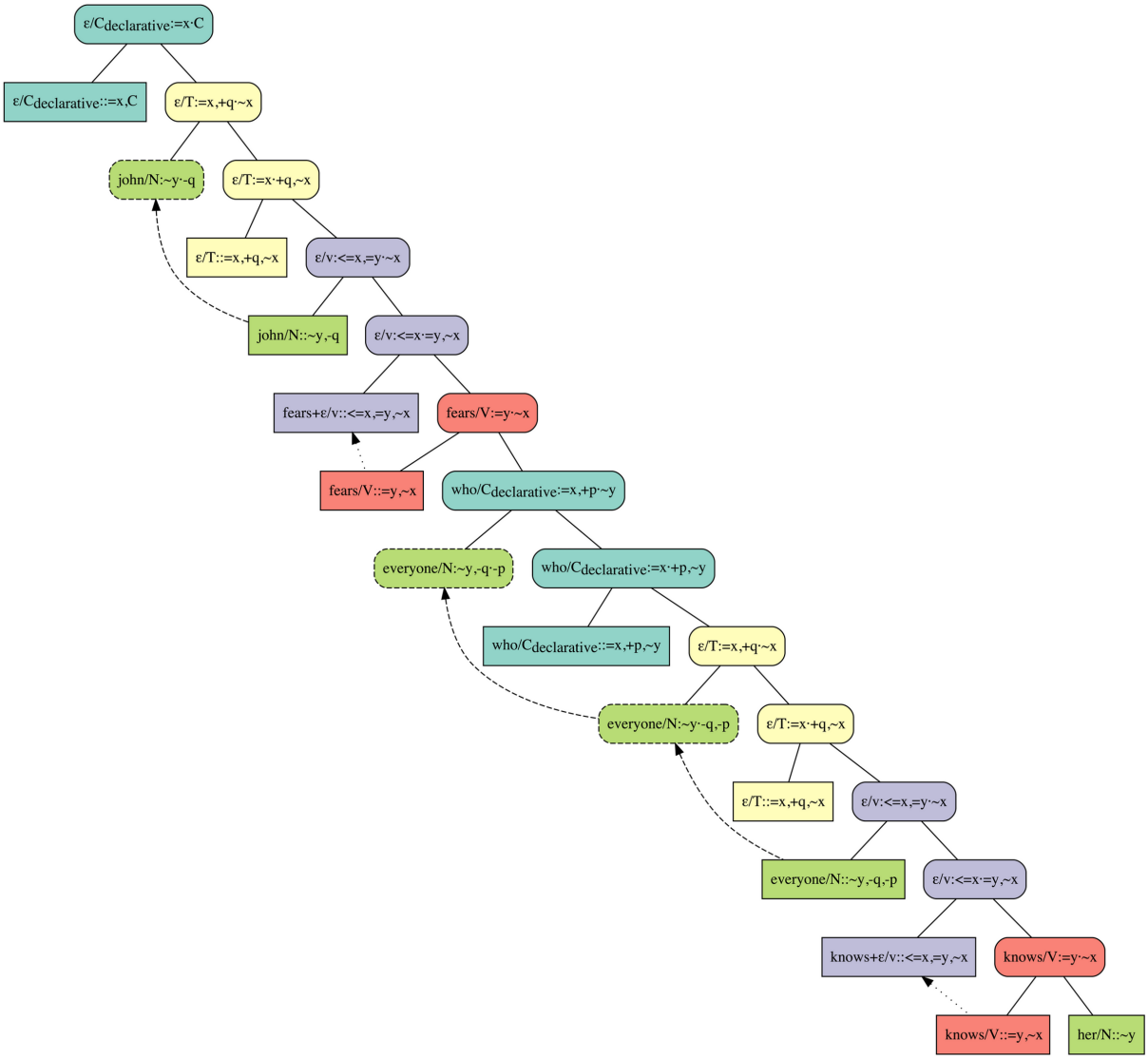

Figure 8: A derivation for the sentence: *"John fears everyone who knows her."* This derivation was output by the parser when it was applied to entry I_7 in Table 2, using the lexicon in Table 1, and matches the derivation prescribed by contemporary theories of minimalist syntax. This demonstrates the parser's capacity to model an input with a relative clause – i.e. *"everyone who knows her"*.