A Procedure for Inferring a Minimalist Lexicon from an SMT Model of a Language Acquisition Device

Sagar Indurkhya INDURKS@MIT.EDU
Massachusetts Institute of Technology

Editors: François Coste, Faissal Ouardi and Guillaume Rabusseau

Abstract

We introduce a constraint-based procedure for inferring a Minimalist Grammar (MG) that falls within the "Logic Grammar" framework. The procedure, implemented as a working computer program, takes as input an MG lexicon and a sequence of sentences paired with their semantic representation, and outputs an MG lexicon that is a superset of the input lexicon and that yields for each input sentence a syntatic structure encoding the associated semantic representation. The procedure operates by first constructing an SMT model of a language acquisition device that is constrained by the input lexicon and the (sentence, semantic-representation) pairs, and then using an SMT-solver to identify a model-solution in which the lexicon is optimized for parsimony. We show how the procedure can be used to form a computational model of a child language learner, presenting two experiments in which the procedure is used for instantaneous and incremental acquisition of an MG lexicon, and find that the optimal MG lexicons inferred by the procedure yield derivations that agree with the prescriptions of contemporary theories of minimalist syntax.

Keywords: Satisfiability Modulo Theories, Minimalist Grammars, Language Acquisition

1. Introduction

Rayner et al.'s (1988) *Logic Grammar* framework for grammatical inference extends the *Parsing as Deduction* framework (Pereira and Warren, 1983), in which a chart parser is formed by coupling a logical deduction engine (e.g. Prolog) with an axiomatization of a CFG grounded in a fixed lexicon (i.e. a *known quantity*), by treating the lexicon as an *unknown quantity* that is required to parse multiple sentences at once, in the hope that with enough sentences, the constraints imposed by this requirement uniquely determine the lexicon. Recently, (Indurkhya, 2022b) has developed a parser for *Minimalist Grammars* (MGs) that operates within the Parsing as Deduction framework, leveraging recent advances in automatic theorem provers by utilizing a high-performance solver for models constructed using a logic, *Satisfiability Modulo Theories* (SMT).[1] This study introduces a novel procedure for inferring MGs, implemented as a working computer program, that adapts and extends Indurkhya's parser in the same manner that Rayner et al. extended a Parsing as Deduction based parser.[2] The inference procedure, falling within the Logic

[1] An SMT formula is expressed in a predicate logic extended with various background theories – e.g. the theories of uninterpreted functions, arrays, arithmetic, and bit-vectors (McCarthy, 1993; Dutertre and de Moura, 2006; Ranise and Tinelli, 2006; Nieuwenhuis and Oliveras, 2006; Nieuwenhuis et al., 2006; de Moura and Bjørner, 2009).

[2] The inference procedure is implemented in Python and is an (MIT-licensed) open source project that is publicly available at `https://github.com/indurks/mgsmt-inference`.

Grammar framework, uses an automatic theorem prover to "solve for syntax" by answering the question: *"given a sequence of sentences paired with structured representations of meaning, what is the optimal lexicon that yields a syntatic structure for each (sentence, meaning) pair."*

The inference procedure, which is a logic program, may be summarized as follows. The procedure takes as input an MG lexicon (that is possibly empty) and a sequence of (sentence, meaning) pairs referred to as the *Primary Linguistic Data* (PLD); more formally, each entry in the PLD is a *Phonological Form* (PF) paired with a *Logical Form* (LF).[3] The procedure outputs an (inferred) MG lexicon that is a superset of the input lexicon and can yield, for each (PF, LF) pair in the PLD, a syntactic structure that satisfies the *Interface Conditions* (ICs) imposed by the PF and LF – namely the PF ICs constrain the surface form that the structure yields, while the LF ICs constrain what semantic interpretation the structure encodes. The procedure operates by first constructing, for each entry in the PLD, an SMT model of an MG derivation that is constrained by the PF and LF ICs for that entry, and connecting each derivation model to the same SMT model of an MG lexicon, which in turn is partially constrained by the input lexicon; the resulting ensemble of (connected) SMT models forms an *SMT model of a language acquisition device.* The procedure then uses an SMT-solver to check (i.e. solve) the full SMT model – if a satisfiable model interpretation (i.e. solution) exists, the procedure: (i) uses the solver to identify a model interpretation that is *optimal* with respect to metrics that reward finding a small and simple lexicon (detailed in §3.2), and then (ii) automatically extracts the (output) MG lexicon from the identified model interpretation. Note that the SMT models of the derivation and lexicon are the same as those found in (Indurkhya, 2022b) – what is new here is that now: (i) *multiple* derivation models are connected to (and thereby constrain) a single lexicon model, (ii) the lexicon model, which is only partially (or even entirely) unconstrained by the input lexicon, is an *unknown quantity* being solved for, and (iii) the solver identifies the *optimal lexicon.*

This study presents two experiments (in §4) that show how the inference procedure can be used to formulate a computational model of a child language learner that meets the criterion for a "language acquisition device" as prescribed in Chomsky's (1965) *Aspects.*[4] We first show how the procedure can be used to formulate an *instantaneous* model of language acquisition, simulating a child language learner that starts with an empty lexicon, processes each entry in the PLD *simultaneously*, and infers a lexicon that constitutes the final state of the learner's *Knowledge of Language* (Chomsky, 1986); remarkably, when the solver is used to infer a lexicon that is *optimal* with respect to metrics encoding *Economy Conditions*, the output lexicon yields derivations that match those prescribed by contemporary theories of minimalist syntax,[5] whereas the unoptimized lexicon does not – hence, at least in this case,

[3] An LF is a structured encoding of syntactic relations that are relevant to semantic interpretation – e.g. predicate-argument structure (Hornstein, 1995; Lepore and Ludwig, 2002; Fox, 2003; Pietroski, 2021).

[4] N.b. the present study is the cumulation of a body of work in which previously published abstracts have documented work-in-progress (Indurkhya, 2020, 2022a) – in particular, the present study: introduces a formal presentation of the inference procedure; details the metrics used for optimization and the model parameters underlying the experiments; and leverages improvements to the SMT-model of an MG derivation, newly introduced in (Indurkhya, 2022b), that prohibit the generation of predicate-argument structures that do not accord with the *Uniformity of Theta Assignment Hypothesis* (Baker, 1988), thus shrinking the space of model solutions (and thereby lowering the runtime of the inference procedure).

[5] As presented in (Adger, 2003; Hornstein et al., 2005; Radford, 2009; Collins and Stabler, 2016).

solving for a parsimonious lexicon yields one that aligns with the prescriptions of linguistic theory. We next show how the procedure can be used to form an *incremental* model of language acquisition, simulating a child language learner who starts with an empty lexicon, and repeatedly runs the inference procedure, each time consuming a new batch of the PLD and taking as input (and then augmenting) the lexicon output by the prior run, thereby incrementally acquiring Knowledge of Language by building up a lexicon; importantly, this shows how the procedure can (i) be applied to an arbitrarily large PLD by consuming it in batches and only requiring memory proportional to the size of target lexicon and the size of a PLD batch, and (ii) infer a lexicon that can generalize beyond the input by yielding sentences with unbounded levels of embedding. The study concludes (in §5) by discussing how the inference procedure can help linguists in the study of language acquisition.

2. Preliminaries

2.1. Minimalist Grammars

MGs are a well-studied class of formal grammars that are capable of closely modeling the syntactic structures prescribed by minimalist theories of syntax.[6] This section reviews the chain-based algebraic formulation of MGs presented in Stabler and Keenan (2003); the reader should consult Fig. 1(a) and Table 3 to anchor the presentation below.

An MG, G, is defined by a tuple, $(\textsc{Sel}, \textsc{Lic}, \sigma, \textsc{Lex}, \mathbb{M})$, the members of which we will now define in turn. To begin, $\textsc{Sel}$ and $\textsc{Lic}$ are finite, non-empty sets of feature labels for (constituent) *selection* and *licensing* (respectively). The set of syntactic features, $\mathcal{F}$, is defined as the union of five sets: *selector* features, $\{{=}x | x \in \textsc{Sel}\}$;[7] *selectee* features, $\{{\sim}x | x \in \textsc{Sel}\}$; *licensor* features, $\{{+}x | x \in \textsc{Lic}\}$; *licensee* features, $\{{-}x | x \in \textsc{Lic}\}$; and a singleton set with the special feature C that terminates a derivation. Next, σ is a finite, non-empty set of phonological forms, each of which is either *overt* or *covert* (denoted by ϵ) depending on whether the form is pronounced or unpronounced (respectively). A *chain* is a sequence of phonological forms paired with a sequence of features, and the set of *chains* may be defined as $\mathcal{H} = \sigma^+ \times \{::, :\} \times \mathcal{F}^+$, where :: and : denote whether a chain is *lexical* or *derived* (from a *lexical* chain), respectively.[8] Then the lexicon, $\textsc{Lex}$, which is composed of lexical chains, is defined to be a (finite) subset of $\sigma \times \{::\} \times \mathcal{F}^+$ (hence $\textsc{Lex} \subset \mathcal{H}$). Finally, given the set of *expressions*, $\mathcal{E} = \mathcal{H}^+$, the (recursive) binary function *Merge*, with signature $\mathbb{M} : \mathcal{E} \times \mathcal{E} \rightarrow \mathcal{E}$, can combine two expressions to form a new expression.

We now define the two (logically disjoint) subcases of $\mathbb{M}$: *external merge* (EM), which models the syntactic *combination* of disjoint expressions, and *internal merge* (IM), which models the syntactic *movement* of one argument that is a constituent of the other (Berwick et al., 2011). EM and IM are driven by feature *selection* and *licensing* (respectively). Let $f \in \textsc{Sel}$, $g \in \textsc{Lic}$, $\gamma \in \mathcal{F}^*$, $\delta \in \mathcal{F}^+$, and $s, t \in \sigma^+$. Additionally, let $\alpha_1, ..., \alpha_k \in \mathcal{H}$ for $0 \leq k$, and let $\iota_1, ..., \iota_l \in \mathcal{H}$ for $0 \leq l$. Then *EM* is the union of three (disjoint) functions

[6] See (Stabler, 1996; Michaelis, 1998, 2001; Harkema, 2001b; Kobele, 2011; Graf, 2011, 2013).

[7] Additionally, a < or > prefixed before a selector prefix – i.e. "<=" or ">=" – indicates that the selector triggers left or right head movement, respectively. See also (Stabler, 2001).

[8] A *chain* traces the trajectory of a lexical entry through a derivation as driven by projection (via *EM*) and raising (via *IM*).

$\{EM_1, EM_2, EM_3\}$ and IM is the union of two (disjoint) functions $\{IM_1, IM_2\}$, as detailed below. (Note that the symbol $\cdot$ indicates that a chain can either be *lexical* or *derived*.)

$$\frac{[\mathrm{s} :: =f,\ \gamma] \qquad [\mathrm{t} \cdot \sim f],\ \iota_1 ... \iota_l}{[\mathrm{st} : \gamma],\ \iota_1 ... \iota_l}\ \mathrm{EM}_1 \qquad\qquad \frac{[\mathrm{s} : +g,\ \gamma],\ \alpha_1 ... \alpha_{i-1},\ [\mathrm{t} : -g],\ \alpha_{i+1} ... \alpha_k}{[\mathrm{ts} : \gamma],\ \alpha_1 ... \alpha_{i-1}, \alpha_{i+1} ... \alpha_k}\ \mathrm{IM}_1$$

$$\frac{[\mathrm{s} : =f,\ \gamma],\ \alpha_1 ... \alpha_k \qquad [\mathrm{t} \cdot \sim f],\ \iota_1 ... \iota_l}{[\mathrm{ts} : \gamma],\ \alpha_1 ... \alpha_k, \iota_1 ... \iota_l}\ \mathrm{EM}_2 \qquad\qquad \frac{[\mathrm{s} : +g,\ \gamma],\ \alpha_1 ... \alpha_{i-1},\ [\mathrm{t} : -g,\ \delta],\ \alpha_{i+1} ... \alpha_k}{[\mathrm{s} : \gamma],\ \alpha_1 ... \alpha_{i-1},\ [\mathrm{t} : \delta],\ \alpha_{i+1} ... \alpha_k}\ \mathrm{IM}_2$$

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

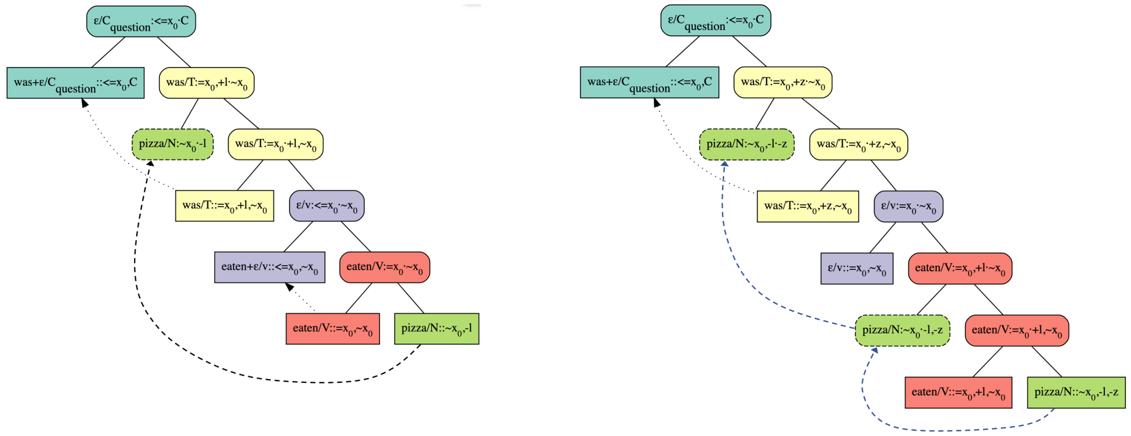
Notably, IM is restricted by the *Shortest Move Constraint* (SMC): a licensor, α, binds to a licensee, β, only if β is the only (available) licensee that α can bind to.[9]

Finally, we define an MG *derivation* as a sequence of expressions generated from a subset of Lex via recursive application of $\mathbb{M}$. A *complete* derivation is one in which the terminal expression consists of a single (derived) chain with one feature, C (which designates the end of the derivation); the terminal expression includes the surface form (i.e. the sequence of phonological forms in the order they would be pronounced). Notably, a derivation, D, appears to take the form of a (conventional) *syntactic structure* if one observes that $\mathbb{M}$ establishes a partial-order over the expressions in D.[10]

Let us now illustrate the MG formalism by stepping through an example of how an MG derivation for the sentence *"Was pizza eaten?"*, shown in Figure 1(*a*), is generated bottom-up using the lexical items listed in Table 3. First, the lexical items for the nominal *"pizza"* and the (lexical) verb *"eaten"* are merged together (via EM_3) to form a verb phrase; note that this application of external merge is allowed because the selector feature $=x_0$ on the term *"eaten"* can check the selectee feature $\sim x_0$ on the term *"pizza"*. This verb phrase is then merged (via EM_1) with the lexical item for a (covert) light verb, ϵ/v, after which the verb *"eaten"* undergoes V-to-v head movement (driven by the presence of the feature $<=x_0$ on the light verb); the resulting term is a (double) VP-shell structure that accords with the Hale-Keyser model of predicate-argument structure (Hale and Keyser, 1993, 2002) – here *"pizza"* is an internal argument of the predicate *"eaten"* (because the former is the complement of the latter). Next, the VP-shell structure is merged with the tense marker *"was"* (via EM_1) and then, in accordance with the VP Internal Subject Hypothesis (Radford, 2009), the (internal) argument *"pizza"* is raised out of the VP-shell structure and merged with *"was"* (via IM_1), after which *"pizza"* is the subject of the sentence; note that this instance of internal merge is driven by the (licensor) feature $+l$ on the term *"was"* licensing the (licensee) feature $-l$ on the term *"pizza"*. Finally, the resulting tense phrase is merged (via EM_1) with the (covert) complementizer, $\epsilon/C_{Ques.}$, after which *"was"* undergoes auxiliary verb raising via T-to-C head-movement (driven by the feature $<=x_0$ on the complementizer) (Pesetsky and Torrego, 2001). This concludes the derivation.

[9] The SMC entails that the licensor will always choose the (hierarchically) closest licensee, since at every stage in a derivation, there can only be one possible licensee that can be licensed – consequently, a derivation is determined entirely by the involved EM operations (Graf, 2013).

[10] N.b. the surface form can be recovered from a syntactic structure by top-down recursive application of a linearization scheme - e.g. for a Subject-Verb-Object language such as English, a specifier-head-complement scheme would be employed (Kayne, 1994).


$\varepsilon/C_{question}:<=x_0\cdot C$
$was+\varepsilon/C_{question}::<=x_0,C$
$was/T:=x_0,+l\cdot\sim x_0$
$pizza/N:\sim x_0\cdot -l$
$was/T:=x_0\cdot +l,\sim x_0$
$was/T::=x_0,+l,\sim x_0$
$\varepsilon/v:<=x_0\cdot\sim x_0$
$eaten+\varepsilon/v::<=x_0,\sim x_0$
$eaten/V:=x_0\cdot\sim x_0$
$eaten/V::=x_0,\sim x_0$
$pizza/N::\sim x_0,-l$
$\varepsilon/C_{question}:<=x_0\cdot C$
$was+\varepsilon/C_{question}::<=x_0,C$
$was/T:=x_0,+z\cdot\sim x_0$
$pizza/N:\sim x_0,-l\cdot -z$
$was/T:=x_0\cdot +z,\sim x_0$
$was/T::=x_0,+z,\sim x_0$
$\varepsilon/v:=x_0\cdot\sim x_0$
$\varepsilon/v::=x_0,\sim x_0$
$eaten/V:=x_0,+l\cdot\sim x_0$
$pizza/N:\sim x_0\cdot -l,-z$
$eaten/V:=x_0\cdot +l,\sim x_0$
$eaten/V::=x_0,+l,\sim x_0$
$pizza/N::\sim x_0,-l,-z$

(*a*) Correct (per theory)

(*b*) Incorrect (violates θ-criterion)

Figure 1: Two MG derivations that satisfy entry I_3 in the PLD (i.e. *"Was pizza eaten?"*); 1(*a*) and 1(*b*) are yielded by the optimized and unoptimized lexicons (respectively) that were inferred in the first experiment (detailed in §4.1). Dashed and dotted arrows mark the movement of phrases and heads (respectively).

Altogether, the capacity of MGs to model syntax has motivated the development of several MG parsers (Harkema, 2001a; Niyogi and Berwick, 2005; Fong and Ginsburg, 2012, 2019; Stabler, 2013; Stanojević, 2016). There has also been prior work on the acquisition of MGs, including earlier models of acquisition grounded in the Minimum Description Length principle (Stabler, 1998), and more recent work in which a wide-coverage MG parser, grounded in a pre-specified MG lexicon, is learned from a treebank of MG derivations using a recurrent neural network (Torr, 2017, 2018; Torr et al., 2019). The present study differentiates itself from this earlier work in so far as the inference procedure can start with an *empty* MG lexicon (i.e. without reference to any pre-specified lexicon), and infer an MG lexicon from a *small* set of sentences paired with semantic representations, each of which do not directly encode a specific MG derivation, as demonstrated by the existence of multiple MG derivations that can satisfy the input ICs (see Fig. 1).

Finally, we take note of prior work that showed how CFGs and multiple context free grammars (MCFGs) can be learned from a distribution of strings (drawn from the target language to be acquired) without the need for structural representations over these strings (e.g. trees or bracketing of substrings) (Clark and Yoshinaka, 2016; Clark and Fijalkow, 2020).[11] The present study is differentiated here in that we explicitly focus on LF ICs being a component of the (input) PLD for two reasons. First, we take a limited formulation of LF ICs to be accessible to a child language learner during the earlier stages of language acquisition – see §3.1 for further discussion. Second, LF ICs impose requirements for specific (strictly) hierarchical relationships, established by *Merge*, to exist within a derivation, and

[11] The MCFG formalism (Seki et al., 1991) is an example of a Mildly Context Sensitive Grammar (Joshi, 1985; Vijay-Shanker et al., 1987; Joshi et al., 1990; Vijay-Shanker and Weir, 1994), which is a class of grammars to which the MG formalism belongs (Michaelis, 1998, 2001).

these requirements are disjoint from the requirements imposed on the derivation by the linear ordering of the surface string encoded in the PF ICs; hence, by partially specifying the input PLD (e.g. only presenting the LF ICs and not the PF ICs), our inference procedure may be used to study the extent to which a grammar can be inferred from the LF ICs alone (see §5 for further discussion).

2.2. Minimalist Parsing using an SMT Solver

We now give an overview of the MGSMT parsing module (Indurkhya, 2022b), which includes an MG parser that this study will extend and adapt to form an inference procedure.

The parser takes as input: (i) an MG lexicon, which consists of a finite set of (word, lexical feature sequence) pairings that serve as the syntatic atoms from which parser can derive a syntactic structure; (ii) a (partial) specification of *Interface Conditions* (ICs) for *Logical Form* (LF) and *Phonological Form* (PF) that serve to restrict what LFs and PFs may be encoded in the syntactic structure that the parser will output. (See Tables 1 and 2 for examples of (PF, LF) ICs.) More specifically:

- The PF ICs consist of a tokenized sentence, with some tokens assigned a category (e.g. "pizza/N").
- The LF ICs consists of: (i) locality constraints – i.e. local hierarchical relations to be established by $\mathbb{M}$ – that encode agreement and predicate-argument structure (Hale and Keyser, 1993, 2002; Chomsky, 2001), and (ii) an indication of whether the sentence is a declarative or an interrogative.[12]

The parser outputs the set of MG derivations (i.e. syntactic structures) that both satisfy the (input) ICs and may be generated from the (input) lexicon.

The parser, a logic program, operates within the Parsing as Deduction framework (Pereira and Warren, 1983; Shieber et al., 1995) and uses an SMT-solver to identify an MG derivation (the *unknown* quantity) that satisfies constraints imposed by the input lexicon and ICs (which are *known* quantities). The parser operates by first constructing an SMT model of an MG lexicon and an SMT model of an MG derivation, and then connecting together the two models (via an uninterpreted function, μ) to form an SMT model of an MG parser.[13][14] Next, the parser converts the inputs into constraints, expressed as SMT-formulae, that augment the SMT model and serve to restrict the space of model solutions – e.g. the input lexicon is translated into constraints that effectively stipulate valuations of the uninterpreted functions in the lexicon model, in order that the lexicon model encode precisely the input lexicon. Finally, the parser produces its output by using an SMT-solver to check whether the SMT model is satisfiable – if it is, the SMT-solver yields (satisfiable) model-interpretations from which the parser extracts the (output) set of MG derivations.

[12] N.b. the LF ICs do not encode information pertaining to the linear ordering of the words in the sentence, they only encode constraints over *hierarchical* relations.

[13] The SMT models are composed of: free finite sorts that stand for domain-objects such as lexical features, phonological forms, categories, expressions in a derivation, etc; uninterpreted (free) functions that encode relationships between domain-objects; model axioms (i.e. SMT-formulae) expressed using a propositional logic extended with (quantifier-free) background theories (e.g. the theory of uninterpreted functions) – these axioms serve to constrain the interpretations of model functions. See (Indurkhya, 2022b) for details.

[14] See both Fig. 3 and §5 in (Indurkhya, 2022b) for specific details of how μ establishes this connection.

Notably, since the parser is a logic program, it can be operated even if the inputs are only partially specified – e.g. Indurkhya (2022b) presents experiments in which the parser is run without any (input) PF ICs and yields a different derivation than when the PF ICs are included in the input. As detailed in §3 below, we make use of the parser's operational flexibility by (i) adapting the parser to be run with the input lexicon only partially restricting the SMT model of the lexicon – i.e. the lexicon model must encode a lexicon that is a *superset* of the input lexicon – and (ii) extending the parser so that multiple derivation models can be connected to (and thus constrain interpretations of) the single lexicon model.

Finally, we note that the method utilized in the present study, in which grammatical inference is cast as an SMT problem, has earlier been successfully applied by Smetsers et al. (2018) for learning a finite state automata (FSA): they encoded an FSA with n states into an SMT model, constrained the model by the strings that the automata must recognize, and then used an SMT solver to check the model and extract the (inferred) FSA.[15] The present study shows that this method for grammatical inference can be used to infer an MG, and taken together with the work of Smetsers et al., suggests that this method is applicable to a broad class of grammatical formalisms spanning multiple levels of the Chomsky hierarchy.

3. Procedure for Inferring an MG Lexicon using an SMT Solver

This section walks through the inference procedure, which is formally detailed in **Alg. 1**.

3.1. Specification of Inputs and Outputs

The procedure takes as input:

1. **Primary Linguistic Data (PLD)**: a sequence of paired PF and LF ICs; the PLD may be divided into batches for incremental processing.
2. **Lexicon**: a list of lexical entries, each entry pairing a phonological form with a feature-sequence – if a lexicon is not provided then the input lexicon is assumed empty.
3. **Model Parameters**: these parameters bound the size of the SMT models to be constructed – this includes bounds on the number of: covert lexical items that can participate in a derivation; instances of phrasal and head movement (respectively) in a derivation; syntactic features in a lexical item; lexical entries associated with each distinct overt phonological form; overt and covert (respectively) entries in the lexicon; distinct selectional and licensing feature labels appearing in the lexicon. Note that bounding the size of the SMT models restricts the (maximum) size of the MG lexicon that can be inferred by the procedure.[16]

The procedure's input takes the form of a JSON dictionary, with key-value pairings for each of the PLD, the lexicon, and model parameters.[17] The procedure outputs an (inferred) MG

[15] See also (Smetsers, 2017).

[16] In practice this does not pose an issue because, as detailed in §3.2, the inference procedure optimizes for the smallest lexicon, so as long as the parameters are large enough to model the smallest lexicon, the procedure will converge on the smallest lexicon. See Indurkhya (2022b, §5) and Indurkhya (2021, Ch. 3) for further details on how model parameters govern the size of the constructed SMT models.

[17] E.g. the JSON syntax for PLD entries I_3 and I_{32} is listed in Appendix-B. Note that as the inference procedure extends and adapts the MGSMT parser, the lexicon and each *individual entry* in the PLD is specified using the same formatting as used by the MGSMT parser – see (Indurkhya, 2022b) for details.

lexicon and, for each entry in the PLD, an MG derivation (generated using the output lexicon) that satisfies the PF and LF ICs listed in that entry; the size of both the output lexicon and the derivations it generates are bound by the (input) model parameters. Importantly, the output lexicon is serialized in the same JSON format as the input lexicon so that it can be consumed as an input in a subsequent run of the inference procedure.[18]

Let us now briefly review the rationale for the PLD entries including the information encoded within the LF ICs.[19] First, association of specific phonological forms with the category of *noun* or *(lexical) verb* is based on evidence that even at the earliest stages of (if not prior to) the acquisition of syntax, the learner attends to assigning categories to the words they hear – e.g. in the case of nouns, see the work by Waxman and Markow (1995) that details how presenting an object with its label focuses the attention of the learner on assigning a category to the object.[20][21] Next, inclusion of predicate-argument structure (for each lexical verb) is based on the *Semantic Bootstrapping Hypothesis*: the argument slots linked to a verb are grounded in the learner's understanding of the meaning of the verb (Grimshaw, 1981; Gleitman, 1990; Pinker, 2009). Finally, inclusion of subject-verb agreement is based on studies showing that, early in the acquisition process, a child language learner: (i) acquires knowledge of how auxiliary verbs are inflected in relation to the subject of the sentence, and (ii) understands that the auxiliary verb (serving as a tense marker) is distinguished from a lexical verb, both in inflection and in the formation of polar interrogatives via (subject-auxiliary) inversion (Lust, 2006, pp. 202–206).

3.2. Constructing and Optimizing the SMT Model

Broadly, the inference procedure can be divided into two consecutive stages – both stages use the *Z3 SMT-solver* (v4.8.7), a high-performance automatic theorem prover that provides facilities for constructing an SMT model, checking (i.e. solving) whether the model is satisfiable, and extracting a satisfiable model-interpretation (if one exists) (de Moura and Bjørner, 2008; Bjørner, 2011).[22][23]

The first stage (i.e. steps 2-3 in Alg. 1) involves constructing an SMT model of the language acquisition device (LAD) and augmenting it with model constraints derived from the inputs (i.e. the PLD and the input lexicon). Specifically, the SMT model of the LAD consists of an SMT model of a lexicon connected to one or more SMT models of a derivation (one for each entry in the input PLD);[24] the lexicon model is augmented with constraints

[18] We augmented the MGSMT toolkit with routines for displaying within a Jupyter notebook (that runs the inference procedure): the PLD and the inferred MG lexicon, as shown in Tables 2 and 3 (respectively), using LaTeX; the MG derivations, as shown in Fig. 1(a), using GraphViz.

[19] The inclusion of any of the information in the LF ICs or the PF ICs is optional – see §5 for a discussion of future experiments in which some of this information might be omitted.

[20] See also Xu et al. (2005); Xu (2007).

[21] See Landau and Gleitman (1985) for a discussion of how a learner may initially distinguish *(lexical) verbs* by leveraging simple conjectures about how verbs encode actions. See also Pinker (2009).

[22] Specifically, *checking the SMT model* involves using the SMT-solver to decide whether the conjunction of the terms on the solver's stack (i.e. S in Alg. 1) is a satisfiable SMT formula.

[23] The inference procedure uses the Python API for Z3 – see `https://github.com/Z3Prover/z3#python`.

[24] Importantly, similar to how the MGSMT parser's SMT model connects the lexicon model to a (single) derivation model via the function, μ, the SMT model of the LAD connects a (single) lexicon model to n distinct derivation models using uninterpreted functions, $\mu_1, \ldots, \mu_n$. (See also step 3f in Alg. 1.)

Algorithm 1: The Inference Procedure

```
1. The input consists of:
    (a) a queue of pairs of interface conditions, referred to as the PLD, with $n > 0$ entries;
    (b) a valuation of model parameters;
    (c) an *empty* SMT-solver stack, $S$, with each entry on the stack to be an SMT-formula, and the conjunction of the entries on the stack to be referred to as "the acquisition model."
    (d) (optional) an initial MG lexicon;
2. The initial state of the procedure, prior to consuming the PLD, is either the lexicon supplied in the input (if one was), or otherwise an empty lexicon:
    (a) initialize a lexicon model (an SMT formula), $m_l$, from the (input) model parameters, the PLD, and the initial lexicon (if one was supplied);
    (b) push $m_l$ onto $S$.
3. The procedure constrains the lexicon model by iteratively consuming the PLD (until it is empty) – on the $i^{th}$ iteration, the procedure will:
    (a) pop entry $I_i$ off of the (PLD) queue;
    (b) initialize a derivation model (an SMT formula), $m_d^i$, from the model parameters and $I_i$;
    (c) push $m_d^i$ onto $S$;
    (d) translate $I_i$ into an SMT-formula, $m_I^i$, that constrains the derivation model $m_d^i$;
    (e) push $m_I^i$ onto $S$;
    (f) construct an SMT-formula, $m_b^i$, that connects the derivation model, $m_d^i$, to the lexicon model, $m_l$, via an uninterpreted function, $\mu_i$;
    (g) push $m_b^i$ onto $S$.
4. The procedure selects a grammar by optimizing the acquisition model:
    (a) for each of METRICS A-D, identify the optimal value for metric $X$ via a (decreasing) linear scan starting at the metric's upper-bound (see §3.2). At each step of the scan:
        i. push onto $S$ an SMT-formula, $c_v^X$, that requires metric $X$ be at most $v$;
        ii. check if the acquisition model is satisfiable using the SMT-solver – if not, pop $c_v^X$ off $S$ and terminate the linear scan (the optimal value for metric $X$ is $v + 1$).
    (b) check the acquisition model using the SMT-solver – if the model is found to be satisfiable, recover the identified (satisfiable) model interpretation (i.e. solution).
5. The output of the procedure consists of:
    (a) for each entry $I_i \in$ PLD, an MG derivation, $d_i$, that satisfies conditions imposed by $I_i$;
    (b) the inferred MG lexicon that can yield each $d_i$;
    (c) the solver stack, $S$, which holds: $m_l$; $m_d^i$ and $m_I^i$ for $1 \leq i \leq n$; optimal values for each optimization metric.
```

requiring that it encode a superset of the input lexicon, and each connected derivation model is augmented with constraints derived from the associated PLD entry's PF and LF ICs (just as the MGSMT parser does). Importantly, the net effect of adding these constraints is a requirement that a satisfiable interpretation (i.e. solution) for the SMT model of the LAD must encode an MG lexicon that can be used to parse each entry in the PLD.

The second stage (i.e. step 4 in Alg. 1) involves *sequentially* optimizing the SMT model of the LAD with respect to each of the following metrics in the order listed below – these metrics are grounded in *Economy Conditions* (Collins, 2001; Lasnik, 2002), which are central principles of the Minimalist Program (Chomsky, 1995).

Metrics

A. This metric counts the number of *distinct* lexical feature sequences (in the lexicon model) that are used by any (connected) derivation, and is bounded above by model parameters that limit the number of distinct lexical feature sequences that the lexicon may have.

B. This metric sums over the number of features in each lexical feature sequence (in the lexicon model) that is used in any (connected) derivation, and is bounded above by the product of: (i) the maximum number of lexical entries the lexicon may have, and (ii) the maximum number of syntactic features a lexical entry may have.

C. This metric counts the total number of (EM and IM) *Merge* operations across the set of derivation models that are connected to the lexicon model. This metric is bounded above by the product of model parameters for: (i) the maximum length of a lexical feature sequence; (ii) the maximum number of leaf nodes a derivation tree may have.

D. This metric counts the number of *distinct* selectional and licensing feature *labels* that appear in the output lexicon, and is bounded above by the number of (distinct) feature labels specified in the model parameters.

Optimizing with respect to metrics A and B serves to reduce the size of the lexicon. Optimizing with respect to metric C serves to simultaneously make every derivation connected to the lexicon as economical as possible by minimizing the total number of *Merge* operations occurring over all of the derivations yielded by the inferred lexicon to satisfy the input PLD (e.g. by reducing instances of syntactic movement). Optimizing with respect to metric D serves to reduce the number of distinct symbols (i.e. feature labels) used by the lexicon, and from an information theoretic standpoint, aims at minimizing the number of bits required to represent the lexicon.[25]

Finally, we observe that since the SMT model is finite and thus all of the SMT formulae in the model can be explicitly quantified, checking the model is a *decidable problem* and the solver is thus guaranteed to eventually identify whether a solution exists.

4. Experiments

The experiments in this section serve as case studies that show how the inference procedure presented in §3 can be used to formulate a *computational model of a child language learner*.[26] This model of a child language learner takes the form prescribed by (Berwick, 1985) and comports with the criterion for a *"language acquisition device"* (LAD) as set out in Chomsky (1965). Accordingly, in each experiment: the initial state of the learner's knowledge (prior to consuming the PLD) consists of an empty lexicon and an axiomatization of minimalist

[25] As a consequence of the Shortest Move Constraint (SMC), the number of distinct licensing features in the lexicon limits the number of crossing dependencies that can overlap one another simultaneously.

[26] Our (Python) implementation of the inference procedure is accompanied by Jupyter Notebooks that replicate the experiments in this study.

syntax that is included in each SMT model of a derivation; the target state of the learner's knowledge (i.e. the acquired Knowledge of Language) is the MG lexicon output by the inference procedure; the inference procedure is used to drive the state of the learner's knowledge from the initial state to the target state by (i) incrementally consuming the input PLD and constructing an SMT model of a LAD that is constrained by the PLD, and then (ii) solving the SMT model and recovering the output lexicon from the solution.[27] Crucially, the (inferred) output lexicon can, for each entry in the PLD, generate an MG derivation that satisfies the LF and PF ICs specified in that entry.

Table 1: Abbreviated presentation of the first batch of the Primary Linguistic Data (PLD) – see Table 6 in Appendix C for the full listing. Within PF ICs, a slash "/" denotes association of a token with a pre-specified category. Within LF ICs: *Agr* denotes agreement; predicate-argument structure is denoted by the θ grid with the subject, object and indirect object denoted by "s:", "o:" and "i:" (respectively); end of sentence punctuation denotes if a sentence is a declarative or an interrogative. Note that each argument of a predicate consist of *multi-set* of phonological forms, not a linear sequence of phonological forms.

I_i	PF Interface Conditions	LF Interface Conditions
I_0	who has eaten/V icecream/N?	$\theta_{\text{eaten}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{has}}[s\colon \text{who}]$
I_1	icecream/N was eaten/V.	$\theta_{\text{eaten}}[o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{icecream}]$
I_2	who was eating/V icecream/N?	$\theta_{\text{eating}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{who}]$
I_3	was pizza/N eaten/V?	$\theta_{\text{eaten}}[o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{pizza}]$
...		
I_{15}	the boy/N has slept/V.	$\theta_{\text{slept}}[s\colon \text{the boy}]$, $Agr_{\text{has}}[s\colon \text{the boy}]$
...		
I_{26}	john/N has told/V mary/N a story/N.	$\theta_{\text{told}}[s\colon \text{john}, o\colon \text{a story}, i\colon \text{mary}]$, $Agr_{\text{has}}[s\colon \text{john}]$
I_{27}	the story/N was told/V to a boy/N.	$\theta_{\text{told}}[o\colon \text{the story}, i\colon \text{to a boy}]$, $Agr_{\text{was}}[s\colon \text{the story}]$
I_{28}	what was john/N asking/V?	$\theta_{\text{asking}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{john}]$

4.1. Experiment 1: Instantaneous Model of Acquisition

In this experiment, the learner's initial state of knowledge is an empty lexicon. The learner consumes the (first) batch of the PLD (see Table 1), which consists of 29 pairings of (PF, LF) ICs – these sentences include declaratives (e.g. I_{26}), polar interrogatives (i.e. yes/no-questions such as I_3), and wh-questions (e.g. I_{28}), in both active voice (e.g. I_2) and passive voice (e.g. I_1) using verbs with different valencies.[28]

The learner then moves from their initial state of knowledge to their (final) target state of knowledge by running the inference procedure, which takes the empty lexicon and the PLD as input, and outputs the (inferred) lexicon shown in Table 3.[29] The (final) inferred lexicon aligns with contemporary theories of minimalist syntax in so far as the lexicon yields the prescribed derivations for a variety of syntactic structures (see Fig. 1(*a*)), utilizing syntactic

[27] Appendix-A details, for each experiment, the values of (input) model parameters and the optimal metric values identified by the SMT-solver.

[28] E.g. see I_{15}, I_0 and I_{27} for instances of intransitive, transitive and ditransitive verbs, respectively.

[29] Specifically, the procedure outputs the subset of the lexicon corresponding to entries marked by a 1.

Table 2: The second, third and fourth batches of the Primary Linguistic Data (PLD).

Batch	I_i	Interface	Interface Conditions
2	I_{29}	PF	john/N has asked/V whether pizza/N was eaten/V.
		LF	$\theta_{\text{asked}}[s\colon \text{john}, o\colon \text{whether pizza was eaten}]$, $Agr_{\text{has}}[s\colon \text{john}]$, $\theta_{\text{eaten}}[o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{pizza}]$
	I_{30}	PF	mary/N was told/V that john/N has eaten/V pizza/N.
		LF	$\theta_{\text{told}}[o\colon \text{that john has eaten pizza}, i\colon \text{mary}]$, $Agr_{\text{was}}[s\colon \text{mary}]$, $\theta_{\text{eaten}}[s\colon \text{john}, o\colon \text{pizza}]$, $Agr_{\text{has}}[s\colon \text{john}]$
	I_{31}	PF	mary/N has told/V john/N that icecream/N was eaten/V.
		LF	$\theta_{\text{told}}[s\colon \text{mary}, o\colon \text{that icecream was eaten}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$, $\theta_{\text{eaten}}[o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{icecream}]$
	I_{32}	PF	mary/N has asked/V john/N whether she/N was eating/V pizza/N.
		LF	$\theta_{\text{asked}}[s\colon \text{mary}, o\colon \text{whether she was eating pizza}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$, $\theta_{\text{eating}}[s\colon \text{she}, o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{she}]$
	I_{33}	PF	who has mary/N told/V that she/N was eating/V icecream/N?
		LF	$\theta_{\text{told}}[s\colon \text{mary}, o\colon \text{that she was eating icecream}, i\colon \text{who}]$, $Agr_{\text{has}}[s\colon \text{mary}]$, $\theta_{\text{eating}}[s\colon \text{she}, o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{she}]$
	I_{34}	PF	who was asked/V whether mary/N has given/V john/N money/N?
		LF	$\theta_{\text{asked}}[o\colon \text{whether mary has given john money}, i\colon \text{who}]$, $Agr_{\text{was}}[s\colon \text{who}]$, $\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{money}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$
3	I_{35}	PF	who has told/V john/N everything/N that mary/N was asked/V?
		LF	$\theta_{\text{told}}[s\colon \text{who}, o\colon \text{everything that mary was asked}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{who}]$, $\theta_{\text{asked}}[o\colon \text{everything}, i\colon \text{mary}]$, $Agr_{\text{was}}[s\colon \text{mary}]$
	I_{36}	PF	was someone/N given/V everything/N that she/N has eaten/V?
		LF	$\theta_{\text{given}}[o\colon \text{everything that she has eaten}, i\colon \text{someone}]$, $Agr_{\text{was}}[s\colon \text{someone}]$, $\theta_{\text{eaten}}[s\colon \text{she}, o\colon \text{everything}]$, $Agr_{\text{has}}[s\colon \text{she}]$
4	I_{37}	PF	mary/N has seen/V everyone/N who john/N was eating/V.
		LF	$\theta_{\text{seen}}[s\colon \text{mary}, o\colon \text{everyone who john was eating}]$, $Agr_{\text{has}}[s\colon \text{mary}]$, $\theta_{\text{eating}}[s\colon \text{john}, o\colon \text{everyone}]$, $Agr_{\text{was}}[s\colon \text{john}]$
	I_{38}	PF	john/N has seen/V someone/N who was eating/V icecream/N.
		LF	$\theta_{\text{seen}}[s\colon \text{john}, o\colon \text{someone who was eating icecream}]$, $Agr_{\text{has}}[s\colon \text{john}]$, $\theta_{\text{eating}}[s\colon \text{someone}, o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{someone}]$
	I_{39}	PF	john/N has seen/V someone/N who was eaten/V.
		LF	$\theta_{\text{seen}}[s\colon \text{john}, o\colon \text{someone who was eaten}]$, $Agr_{\text{has}}[s\colon \text{john}]$, $\theta_{\text{eaten}}[o\colon \text{someone}]$, $Agr_{\text{was}}[s\colon \text{someone}]$

Table 3: A *factored* view of the inferred lexicon output by the inference procedure in the course of the experiments detailed in §4. Each row indicates the phonological forms that are paired with the listed category and lexical feature sequence (LFS) in the lexicon. The number in an entry codes for the PLD batch that was processed when the associated (phonological form, LFS) pair first entered into the lexicon (a · denotes an empty entry).

ID	Category	Features	Phonological Forms																															
			what	who	that	whether	has	was	she	everything	someone	icecream	money	mary	pizza	john	nothing	boy	story	given	told	asked	known	seen	asking	eating	eaten	to	everyone	sleeping	slept	the	a	ϵ
$\mathfrak{L}_1$	V	$= x_0, \sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1	1	4	1	1	1	·	·	·	·	·	·	·
$\mathfrak{L}_2$	V	$= x_0, = x_0, \sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1	1	1	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_3$	$C_{decl.}$	$= x_0, C$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1
$\mathfrak{L}_4$	$C_{ques.}$	$<= x_0, +z, C$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1
$\mathfrak{L}_5$	v	$<= x_0, \sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1
$\mathfrak{L}_6$	$C_{ques.}$	$<= x_0, C$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1
$\mathfrak{L}_7$	v	$<= x_0, = x_0, \sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1
$\mathfrak{L}_8$	P	$= x_0, \sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·
$\mathfrak{L}_9$	D	$= x_0, \sim x_0, -l$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1	1	·
$\mathfrak{L}_{10}$	D	$= x_0, \sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1	1	·
$\mathfrak{L}_{11}$	D	$\sim x_0, -z$	1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{12}$	D	$\sim x_0, -l, -z$	1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{13}$	T	$= x_0, +l, \sim x_0$	·	·	·	·	1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{14}$	V	$\sim x_0$	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	1	1	·	·	·
$\mathfrak{L}_{15}$	N	$\sim x_0, -l$	·	·	·	·	·	·	2	1	1	1	1	1	1	1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{16}$	N	$\sim x_0$	·	·	·	·	·	·	·	1	1	1	1	1	1	1	1	1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{17}$	$C_{decl.}$	$= x_0, \sim x_0$	·	·	2	2	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{18}$	$C_{decl.}$	$= x_0, +z, \sim x_0$	·	4	3	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
$\mathfrak{L}_{19}$	N	$\sim x_0, -z$	·	·	·	·	·	·	·	3	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	·	·	·	·	·
$\mathfrak{L}_{20}$	N	$\sim x_0, -l, -z$	·	·	·	·	·	·	·	·	4	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

movement such as *wh*-raising (e.g. $\mathfrak{L}_4$ and $\mathfrak{L}_{11}$) and subject-raising (e.g. $\mathfrak{L}_{13}$ and $\mathfrak{L}_{15}$), head-movement such as T-to-C (e.g. $\mathfrak{L}_6$ with $\mathfrak{L}_{13}$) and V-to-v (e.g. $\mathfrak{L}_5$ with $\mathfrak{L}_1$), and covert lexical items serving as complementizers (e.g. $\mathfrak{L}_3$) and light-verbs (e.g. $\mathfrak{L}_5$), as needed.[30]

This illustrates how the inference procedure can be used to form an *instantaneous* model of language acquisition, so called because it simulates a learner that has to consume the entire PLD before inferring a lexicon (Lust, 2006). More specifically, the SMT models constructed by the procedure, which consists of the lexicon model and the 29 connected derivation models (one for each entry in the input PLD), must be solved by the SMT-solver simultaneously. The *instantaneous* model of acquisition is advantageous in so far as the procedure is guaranteed to infer the optimal MG lexicon no matter the order in which the PLD is presented to the learner, so long as such a lexicon that is compatible with that PLD exists and the (finite) constructed model is large enough to represent it; however, it has a big disadvantage in that the size of the constructed SMT model scales with the size of the input PLD (since a derivation model must be instantiated for each entry in the PLD), and consequently the runtime of the SMT-solver grows intractable as the PLD gets larger.

4.2. Experiment 2: Incremental Model of Acquisition

In this second experiment, the learner's initial state of knowledge is again an empty lexicon. However, the learner will now *incrementally* consume the PLD in four successive batches, with the first batch listed in Table 1 (the same batch consumed in the first experiment) and the remaining three batches listed in Table 2, making for a total of 40 pairings of (PF, LF) ICs. Notably, this PLD goes beyond the PLD consumed in the first experiment in so far as it includes degree-1 embedding constructions: the second batch introduces sentences with an embedded clause that is either an interrogative (e.g. I_{32}) or a declarative (e.g. I_{33}), while the third and fourth batch introduce sentences in which the embedded clause is a (restrictive) relative clause (e.g. I_{38}).

The learner moves from their initial state of knowledge to the (final) target state of knowledge by repeatedly running the inference procedure, each time taking as input the next batch of the PLD and the lexicon output by the prior run of the procedure; importantly, each time the inference procedure is run, the lexicon that is output is a superset of the lexicon that was input, and in this way the learner is using the procedure to incrementally grow a lexicon (that encodes their state of knowledge).

Table 3 shows the final lexicon acquired by the learner as well as the subset of the lexicon learned after processing each specific batch of the PLD. In processing batches 2-4, the procedure augmented the lexicon with entries that: model raising an antecedent that originates within a relative clause (e.g. $\mathfrak{L}_{18}$ and $\mathfrak{L}_{20}$); pair new phonological forms with already known lexical feature sequences (e.g. $\mathfrak{L}_{15}$ associating with "she"); pair newly inferred lexical feature sequences with already known phonological forms (e.g. "that" associating with $\mathfrak{L}_{18}$); and pair a new phonological form with a newly inferred lexical feature sequence (e.g. "whether" is paired with $\mathfrak{L}_{17}$). Moreover, the inferred lexicon yields derivations that include

[30] Notably, the (sub-optimal) lexicon inferred by the solver prior to metrics A-D being optimized does not yield derivations that agree with contemporary theories of minimalist syntax. E.g. Fig. 1(*b*) shows a derivation (yielded by the sub-optimal lexicon) in which the argument "pizza" (incorrectly) merges twice with a transitive verb "eaten" – by the *Uniformity of Theta Assignment Hypothesis* (Baker, 1988), "eaten" will assign "pizza" two distinct θ-roles, thereby violating the *Theta Criterion* (Chomsky, 1981).

embedded sentences and (embedded) relative clauses – e.g. Figs. 2 and 3 (in Appendix-D) show derivations for I_{32} and I_{38}, respectively. Finally, the inferred lexicon generalizes beyond the derivations required to satisfy the input PLD – e.g. although the PLD only includes sentences with at most one level of embedding, the final (inferred) lexicon can generate derivations with n-levels of embedding for any $n \geq 0$.[31]

In summary, this experiment shows how an *incremental* theorem prover – e.g. the stack-based Z3 SMT-solver – can be used to acquire a grammar in stages from an arbitrarily large (batched) PLD, while requiring only a small amount of memory since only a single batch of the PLD needs to be processed at a time.

5. Conclusions

We have revisited the Logic Grammar framework in a modern setting grounded in minimalist syntax, extending and adapting the (Parsing as Deduction based) MGSMT parser to form a novel inference procedure for MGs. The procedure, implemented as a working computer program, leverages recent advances in automatic theorem proving: after constructing an SMT model of a language acquisition device, it uses a high-performance SMT-solver to declaratively-deduce an optimal model solution from which a parsimonious MG lexicon is recovered.

The experiments in this study show how the inference procedure can be used to form a psychologically plausible model of language acquisition – remarkably, the recovered (optimal) MG lexicon yields derivations that comport with contemporary theories of minimalist syntax. More broadly, the experiments demonstrate how an SMT-solver can assist the study of language acquisition via a division of labor: a linguist can focus on developing computational experiments by specifying the learner's initial state and the conditions that the learner's final state (i.e. the inferred MG lexicon) should satisfy (as encoded by the PLD), and leave to the solver the task of determining what the learner's final state is and how the language acquisition device drives from the initial state to that final state.

Looking ahead, one avenue of future work involves repeating the experiments in §4 while only partially specifying the PLD – e.g. only the LF ICs would be included in each entry in the PLD; this is possible because the inference procedure is a logic program in which any *known* quantity can be made an *unknown* quantity by deleting the appropriate constraints (e.g. constraints imposed by PF ICs). Such experiments may better our understanding of what Knowledge of Language can be acquired strictly from exposure to semantic representations originating in other mental systems (via the LF interface), and identify knowledge that can only be learned via exposure to the Sensory-Motor system (via the PF interface).

Acknowledgments

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

[31] E.g. given that the inferred lexicon can derive the sentence *"A boy has told someone the story."* using the lexical entries $\{\mathfrak{L}_3, \mathfrak{L}_9, \mathfrak{L}_{16}, \mathfrak{L}_{13}, \mathfrak{L}_7, \mathfrak{L}_2, \mathfrak{L}_{10}\}$, a derivation with $n > 0$ levels of embedding may be produced by n (repeated) applications of a rule that replaces the argument *"the story"* with the (embedded) clause *"that a boy has told someone the story"*, specifically by replacing $\{the/\mathfrak{L}_{10}, story/\mathfrak{L}_{16}\}$ with $\{that/\mathfrak{L}_{17}$, $a/\mathfrak{L}_9$, $boy/\mathfrak{L}_{16}$, $has/\mathfrak{L}_{13}$, $\epsilon_v/\mathfrak{L}_7$, $told/\mathfrak{L}_2$, $someone/\mathfrak{L}_{16}$, $the/\mathfrak{L}_{10}$, $story/\mathfrak{L}_{16}\}$.

References

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

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

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

Robert C. Berwick. *The acquisition of syntactic knowledge.* MIT Press, 1985.

Robert C. Berwick, Paul Pietroski, Beracah Yankama, and Noam Chomsky. Poverty of the stimulus revisited. *Cognitive Science*, 35(7):1207–1242, 2011.

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

Noam Chomsky. *Aspects of the theory of syntax.* MIT Press, 1965.

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

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

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

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

Alexander Clark and Nathanaël Fijalkow. Consistent unsupervised estimators for anchored pcfgs. *Transactions of the Association for Computational Linguistics*, 8:409–422, 2020.

Alexander Clark and Ryo Yoshinaka. *Distributional Learning of Context-Free and Multiple Context-Free Grammars*, pages 143–172. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. doi: 10.1007/978-3-662-48395-4_6.

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

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

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

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

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

Sandiway Fong and Jason Ginsburg. Computation with doubling constituents: Pronouns and antecedents in phase theory. *Towards a Biolinguistic understanding of grammar: Essays on interfaces*, pages 303–338, 2012.

Sandiway Fong and Jason Ginsburg. Towards a minimalist machine. In R. C. Berwick and E. P. Stabler, editors, *Minimalist Parsing*, pages 16–38. Oxford University Press, 2019.

Danny Fox. On logical form. *Minimalist syntax*, pages 82–123, 2003.

Lila Gleitman. The structural sources of verb meanings. *Language acquisition*, 1(1):3–55, 1990.

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

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

Jane Grimshaw. Form, function, and the language acquisition device. *The logical problem of language acquisition*, 165:178, 1981.

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

Kenneth Hale and Samuel J. Keyser. 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, 1993.

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

Hendrik Harkema. *Parsing Minimalist Languages.* PhD thesis, University of California Los Angeles, 2001a.

Henk Harkema. A characterization of minimalist languages. In *Logical Aspects of Computational Linguistics: 4th International Conference, LACL 2001 Le Croisic, France, June 27–29, 2001 Proceedings 4*, pages 193–211. Springer, 2001b.

Norbert Hornstein. *Logical form: From GB to minimalism.* John Wiley & Sons, 1995.

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

Sagar Indurkhya. Inferring minimalist grammars with an SMT-solver. *Proceedings of the Society for Computation in Linguistics*, 3(1):476–479, 2020.

Sagar Indurkhya. *Solving for syntax.* PhD thesis, Massachusetts Institute of Technology, 2021.

Sagar Indurkhya. Incremental acquisition of a Minimalist Grammar using an SMT-solver. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 212–216, online, February 2022a. Association for Computational Linguistics. URL https://aclanthology.org/2022.scil-1.19.

Sagar Indurkhya. Parsing as deduction revisited: Using an automatic theorem prover to solve an SMT model of a minimalist parser. In *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, pages 157–175, Abu Dhabi, United Arab Emirates (Hybrid), December 2022b. Association for Computational Linguistics. URL https://aclanthology.org/2022.conll-1.12.

Aravind K. Joshi. Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description? *Natural Language Parsing*, pages 206–250, 1985.

Aravind K Joshi, Krishnamurti Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. *Technical Reports (CIS)*, page 539, 1990.

Richard S Kayne. *The antisymmetry of syntax.* Number 25 in Linguistic Inquiry Monographs. MIT Press, 1994.

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

Barbara Landau and Lila R Gleitman. *Language and experience: Evidence from the blind child.* Harvard University Press, 1985.

Howard Lasnik. The minimalist program in syntax. *Trends in cognitive sciences*, 6(10): 432–437, 2002.

Ernest Lepore and Kirk Ludwig. What is logical form? In Gerhard Preyer and Georg Peter, editors, *Logical Form and Language*, pages 54–90. Clarendon Press, 2002.

Barbara C Lust. *Child language: Acquisition and growth.* Cambridge University Press, 2006.

John McCarthy. Towards a mathematical science of computation. *Program Verification: Fundamental Issues in Computer Science*, pages 35–56, 1993.

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

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

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

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. 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, 2006.

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

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

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

Paul Pietroski. Logical Form. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

Steven Pinker. *Language learnability and language development: with new commentary by the author*, volume 7. Harvard University Press, 2009.

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

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

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

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991.

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

Rick Smetsers. Grammatical inference as a satisfiability modulo theories problem. *CoRR*, abs/1705.10639, 2017. URL http://arxiv.org/abs/1705.10639.

Rick Smetsers, Paul Fiterau-Brostean, and Frits W. Vaandrager. Model learning as a satisfiability modulo theories problem. In Shmuel Tomi Klein, Carlos Martín-Vide, and Dana Shapira, editors, *Language and Automata Theory and Applications - 12th International Conference, LATA 2018, Ramat Gan, Israel, April 9-11, 2018, Proceedings*, volume 10792 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2018. doi: 10.1007/978-3-319-77313-1_14. URL https://doi.org/10.1007/978-3-319-77313-1_14.

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

Edward P. Stabler. Acquiring languages with movement. *Syntax*, 1(1):72–97, 1998.

Edward P Stabler. Recognizing head movement. In *International Conference on Logical Aspects of Computational Linguistics*, pages 245–260. Springer, 2001.

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

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

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

John Torr. Autobank: a semi-automatic annotation tool for developing deep minimalist grammar treebanks. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–86, 2017.

John Torr. Constraining mgbank: Agreement, l-selection and supertagging in minimalist grammars. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 590–600. Association for Computational Linguistics, 2018.

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

Krishnamurti Vijay-Shanker and David J Weir. The equivalence of four extensions of context-free grammars. *Mathematical systems theory*, 27(6):511–546, 1994.

Krishnamurti Vijay-Shanker, David J. Weir, and Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, ACL '87, pages 104–111, Stroudsburg, PA, USA, 1987. Association for Computational Linguistics.

Sandra R Waxman and Dana B Markow. Words as invitations to form categories: Evidence from 12-to 13-month-old infants. *Cognitive psychology*, 29(3):257–302, 1995.

Fei Xu. Sortal concepts, object individuation, and language. *Trends in Cognitive Sciences*, 11(9):400–406, 2007.

Fei Xu, Melissa Cote, and Allison Baker. Labeling guides object individuation in 12-month-old infants. *Psychological Science*, 16(5):372–377, 2005.

Appendix A. Model Parameters and Metric Valuations for Experiments

Table 4: This table shows the values used for the model parameters that were input into the procedure in each of the two experiments detailed in §4. Here the variable n_p stands for the number of predicates appearing in the LF ICs that constrain a derivation – hence, a derivation with no embedding structure (e.g. the derivation for I_3) will have $n_p = 1$, whereas a derivation with one level of embedding (e.g. I_{32}) will have $n_p \geq 2$; this heuristic for relaxing these model parameters (that serve to bound the constructed SMT models) is grounded in the notion that each clause in a derivation, whether the matrix clause or an embedded clause, involves an *Extended Functional Projection* (of the form $C{-}T{-}v{-}V$) that is anchored in a predicate. Note that an *Extended Functional Projection* is a constraint over the structural arrangements that lexical heads may enter into within a derivation (Grimshaw, 2005; Adger and Svenonius, 2011).

Model Parameter	Value
Max. number of covert lexical items that may participate in a derivation.	$2n_p$
Max. number of instances of phrasal movement (i.e. IM) in a derivation.	$2n_p$
Max. number of instances of head movement in a derivation.	$2n_p$
Max. number of syntactic features in a lexical entry (i.e. the length of a lexical feature sequence).	3
Max. number of lexical entries that may be associated with each distinct *overt* phonological form appearing in the PLD.	3
Max. number of *covert* lexical entries in the lexicon.	5
Max. number of *distinct* selectional feature labels that appear in the lexicon (i.e. cardinality of Sel).	3
Max. number of *distinct* licensing features labels that appear in the lexicon (i.e. cardinality of Lic).	3

Table 5: This table lists, for each PLD batch, the optimal value for each metric (in §3.2) as identified by the SMT-solver when the inference procedure was processing that batch of the PLD; note that the same optimal metric values for the first PLD batch were achieved in both experiments (as expected, since the second experiment extends the first).

PLD Batch	Optimal Metric Values			
	Metric A	Metric B	Metric C	Metric D
1	16	33	425	3
2	17	35	158	3
3	19	40	58	3
4	20	43	79	3

Appendix B. Examples of the JSON Syntax Used to Encode the PLD

```
{
  "sentence": "was pizza eaten ?",
  "locality_constraints": [
    ["theta", {"pred": "eaten", "obj": "pizza"}],
    ["agree", {"subj": "pizza", "pred": "was"}]
  ],
  "categorical_constraints": {
    "N": ["pizza"],
    "V": ["eaten"]
  }
}
```

Listing 1: JSON syntax for PLD entry I_3 as specified in Table 1. The LF interface conditions are listed under the "locality_constraints" key. The PF interface conditions are listed under the "sentence" and "categorical_constraints" keys, with the former encoding a linear sequence of *overt* phonological forms, and the latter associating (*overt*) phonological forms with specific syntactic categories (with the possible categories being C, v, T, V, P, D and N).

```
{
  "sentence": "mary has asked john whether she was eating pizza .",
  "locality_constraints": [
    ["theta", {
      "pred": "asked",
      "subj": "mary",
      "iobj": "john",
      "obj": "whether she was eating pizza"}
    ],
    ["agree", {"subj": "mary", "pred": "has"}],
    ["theta", {"pred": "eating", "subj": "she", "obj": "pizza"}],
    ["agree", {"subj": "she", "pred": "was"}]],
  "categorical_constraints": {
    "N": ["john", "mary", "she", "pizza"],
    "V": ["asked", "eating"]}
}
```

Listing 2: JSON syntax for entry I_{32} as specified in the third batch of the PLD listed in Table 2. Note that when an embedded clause is an argument, the listed phrase is to be interpreted as a *multi-set* of phonological forms – e.g. in I_{32}, the *multi-set* of phonological forms $\{whether, she, was, eating, pizza\}$ serves as an internal argument (obj) of the lexical verb *"asked."*

Appendix C. Complete Listing of the First Batch of the PLD

Table 6: The first batch of the Primary Linguistic Data (PLD). Within PF ICs, a slash "/" denotes association of a token with a pre-specified category. Within LF ICs: *Agr* denotes agreement; predicate-argument structure is denoted by the θ grid with the subject, object and indirect object denoted by "s:", "o:" and "i:" (respectively); end of sentence punctuation denotes if a sentence is a declarative or an interrogative. Note that each argument of a predicate consist of *multi-set* of phonological forms, not a (linear) sequence of phonological forms.

ID	PF Interface Conditions	LF Interface Conditions
I_0	who has eaten/V icecream/N?	$\theta_{\text{eaten}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{has}}[s\colon \text{who}]$
I_1	icecream/N was eaten/V.	$\theta_{\text{eaten}}[o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{icecream}]$
I_2	who was eating/V icecream/N?	$\theta_{\text{eating}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{who}]$
I_3	was pizza/N eaten/V?	$\theta_{\text{eaten}}[o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{pizza}]$
I_4	what has john/N eaten/V?	$\theta_{\text{eaten}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{has}}[s\colon \text{john}]$
I_5	has mary/N eaten/V pizza/N?	$\theta_{\text{eaten}}[s\colon \text{mary}, o\colon \text{pizza}]$, $Agr_{\text{has}}[s\colon \text{mary}]$
I_6	was john/N eating/V pizza/N?	$\theta_{\text{eating}}[s\colon \text{john}, o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{john}]$
I_7	what was mary/N eating/V?	$\theta_{\text{eating}}[s\colon \text{mary}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{mary}]$
I_8	what was eaten/V?	$\theta_{\text{eaten}}[o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{what}]$
I_9	was mary/N given/V pizza/N?	$\theta_{\text{given}}[o\colon \text{pizza}, i\colon \text{mary}]$, $Agr_{\text{was}}[s\colon \text{mary}]$
I_{10}	what has mary/N given/V john/N?	$\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{what}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$
I_{11}	mary/N has given/V john/N money/N.	$\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{money}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$
I_{12}	who was money/N given/V to/P?	$\theta_{\text{given}}[o\colon \text{money}, i\colon \text{to who}]$, $Agr_{\text{was}}[s\colon \text{money}]$
I_{13}	who has john/N given/V money/N to/P?	$\theta_{\text{given}}[s\colon \text{john}, o\colon \text{money}, i\colon \text{to who}]$, $Agr_{\text{has}}[s\colon \text{john}]$
I_{14}	was the boy/N sleeping/V?	$\theta_{\text{sleeping}}[s\colon \text{the boy}]$, $Agr_{\text{was}}[s\colon \text{the boy}]$
I_{15}	the boy/N has slept/V.	$\theta_{\text{slept}}[s\colon \text{the boy}]$, $Agr_{\text{has}}[s\colon \text{the boy}]$
I_{16}	john/N was told/V nothing/N.	$\theta_{\text{told}}[o\colon \text{nothing}, i\colon \text{john}]$, $Agr_{\text{was}}[s\colon \text{john}]$
I_{17}	someone/N has known/V everything/N.	$\theta_{\text{known}}[s\colon \text{someone}, o\colon \text{everything}]$, $Agr_{\text{has}}[s\colon \text{someone}]$
I_{18}	who was asking/V nothing/N?	$\theta_{\text{asking}}[s\colon \text{who}, o\colon \text{nothing}]$, $Agr_{\text{was}}[s\colon \text{who}]$
I_{19}	nothing/N was asked/V.	$\theta_{\text{asked}}[o\colon \text{nothing}]$, $Agr_{\text{was}}[s\colon \text{nothing}]$
I_{20}	everything/N was known/V.	$\theta_{\text{known}}[o\colon \text{everything}]$, $Agr_{\text{was}}[s\colon \text{everything}]$
I_{21}	who was everything/N told/V to?	$\theta_{\text{told}}[o\colon \text{everything}, i\colon \text{to who}]$, $Agr_{\text{was}}[s\colon \text{everything}]$
I_{22}	john/N has asked/V someone/N everything/N.	$\theta_{\text{asked}}[s\colon \text{john}, o\colon \text{everything}, i\colon \text{someone}]$, $Agr_{\text{has}}[s\colon \text{john}]$
I_{23}	what was someone/N asked/V?	$\theta_{\text{asked}}[o\colon \text{what}, i\colon \text{someone}]$, $Agr_{\text{was}}[s\colon \text{someone}]$
I_{24}	who has told/V someone/N the story/N?	$\theta_{\text{told}}[s\colon \text{who}, o\colon \text{the story}, i\colon \text{someone}]$, $Agr_{\text{has}}[s\colon \text{who}]$
I_{25}	a boy/N was eating/V the pizza/N.	$\theta_{\text{eating}}[s\colon \text{a boy}, o\colon \text{the pizza}]$, $Agr_{\text{was}}[s\colon \text{a boy}]$
I_{26}	john/N has told/V mary/N a story/N.	$\theta_{\text{told}}[s\colon \text{john}, o\colon \text{a story}, i\colon \text{mary}]$, $Agr_{\text{has}}[s\colon \text{john}]$
I_{27}	the story/N was told/V to a boy/N.	$\theta_{\text{told}}[o\colon \text{the story}, i\colon \text{to a boy}]$, $Agr_{\text{was}}[s\colon \text{the story}]$
I_{28}	what was john/N asking/V?	$\theta_{\text{asking}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{john}]$

Appendix D. Examples of Derivations Yielded by an Inferred Lexicon

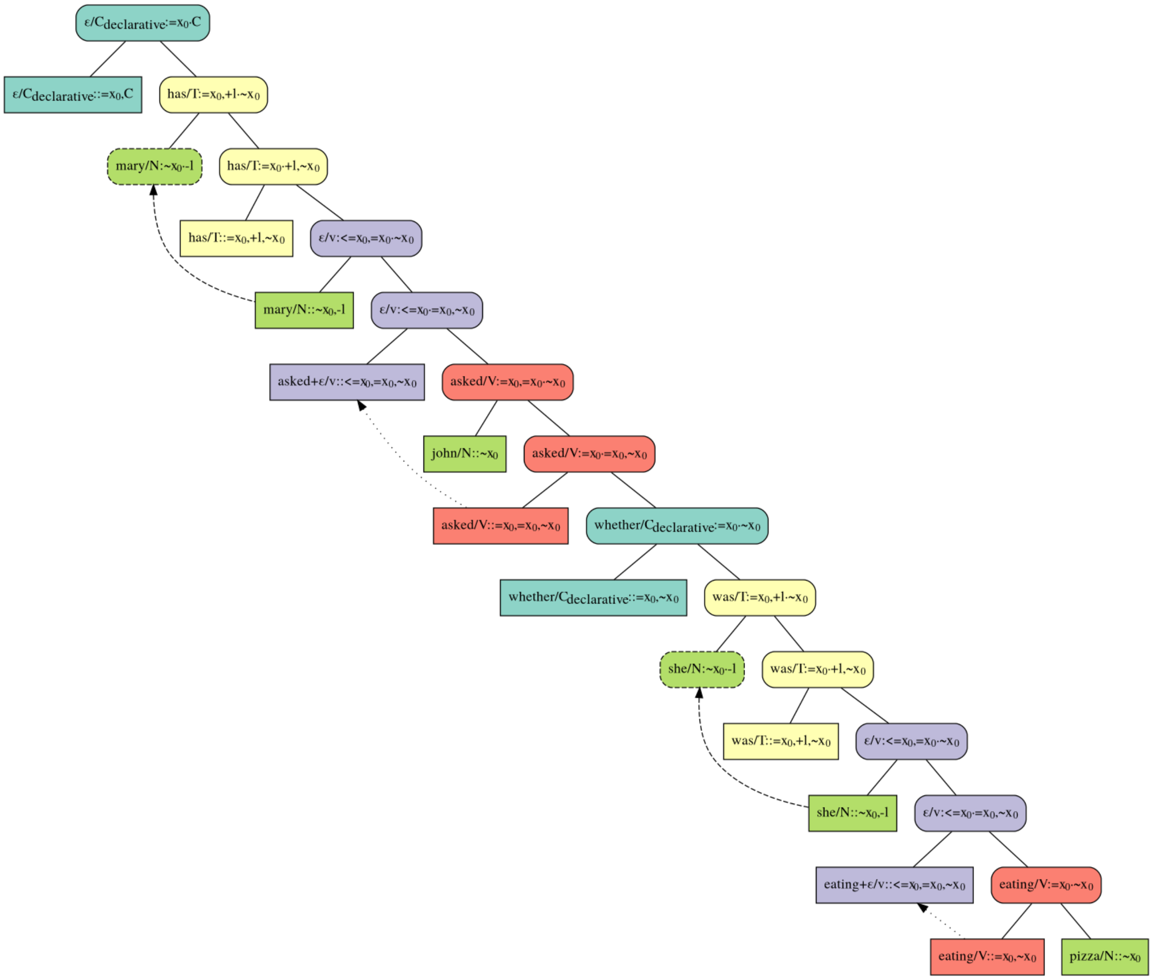


Figure 2: MG derivation yielded by the inferred lexicon (in Table 3) that satisfies the LF and PF interface conditions given in entry I_{32} (i.e. *"Mary has asked John whether she was eating pizza."*) of the PLD listed in Table 2. Dashed and dotted arrows mark the movement of phrases and heads (respectively). This derivation derives a sentence with an embedded question using the lexical feature sequences: $\mathfrak{L}_1$, $\mathfrak{L}_2$, $\mathfrak{L}_3$, $\mathfrak{L}_7$, $\mathfrak{L}_{13}$, $\mathfrak{L}_{15}$, $\mathfrak{L}_{16}$ and $\mathfrak{L}_{17}$.

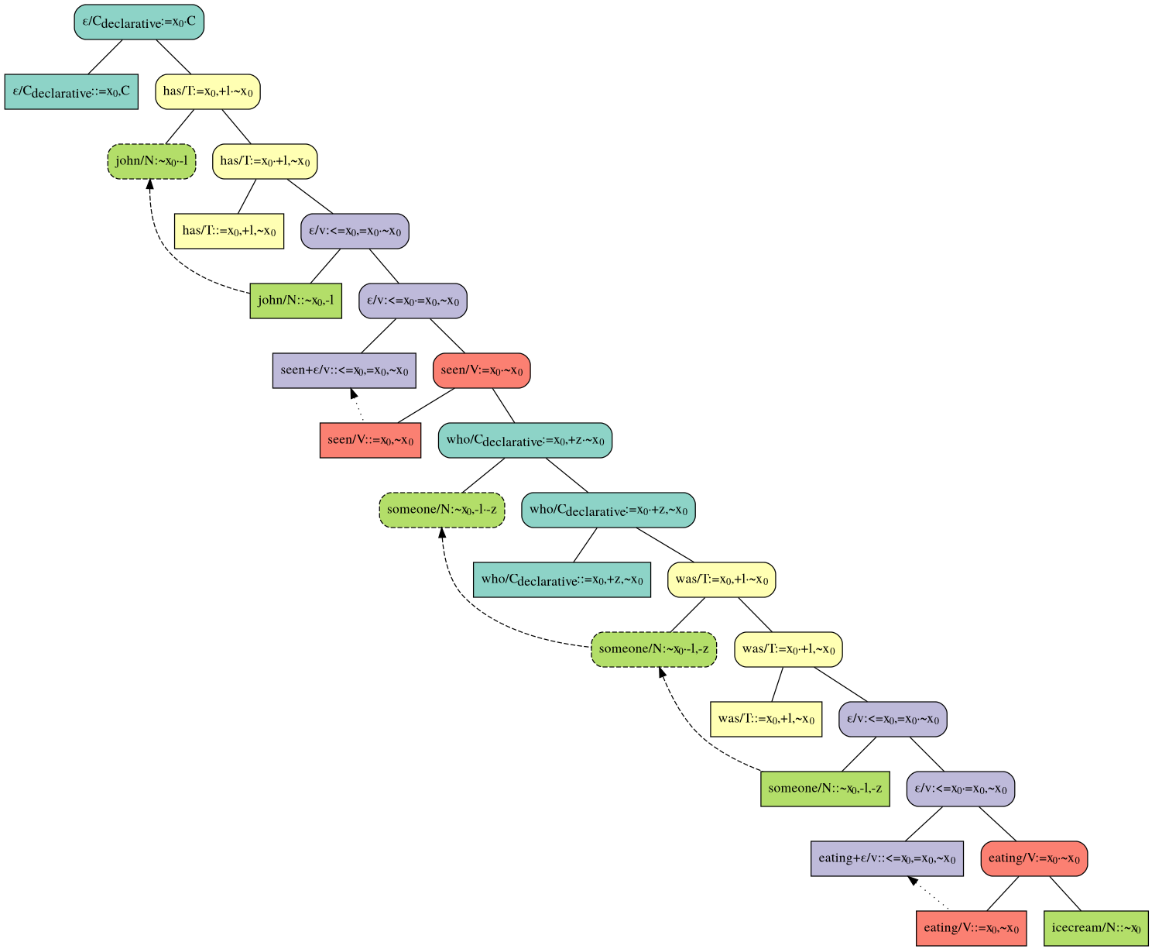


Figure 3: MG derivation yielded by the inferred lexicon (in Table 3) that satisfies the LF and PF interface conditions given in entry I_{38} (i.e. *"John has seen someone who was eating icecream."*) of the PLD listed in Table 2. Dashed and dotted arrows mark the movement of phrases and heads (respectively). This derivation derives a sentence with an embedded (restrictive) relative clause using the lexical feature sequences: $\mathfrak{L}_1$, $\mathfrak{L}_3$, $\mathfrak{L}_7$, $\mathfrak{L}_{13}$, $\mathfrak{L}_{15}$, $\mathfrak{L}_{16}$, $\mathfrak{L}_{18}$ and $\mathfrak{L}_{20}$.