Reinforcement learning of minimalist grammars

Peter beim Graben, Ronald Römer, Werner Meyer, Markus Huber, and Matthias Wolff

Abstract Speech-controlled user interfaces facilitate the operation of devices and household functions to laymen. State-of-the-art language technology scans the acoustically analyzed speech signal for relevant keywords that are subsequently inserted into semantic slots to interpret the user's intent. In order to develop proper cognitive information and communication technologies, simple slot-filling should be replaced by utterance meaning transducers (UMT) that are based on semantic parsers and a *mental lexicon*, comprising syntactic, phonetic and semantic features of the language under consideration. This lexicon must be acquired by a cognitive agent during interaction with its users. We outline a reinforcement learning algorithm for the acquisition of syntax and semantics of English utterances, based on minimalist grammar (MG), a recent computational implementation of generative linguistics. English declarative sentences are presented to the agent by a teacher in form of utterance meaning pairs (UMP) where the meanings are encoded as formulas of predicate logic. Since MG codifies universal linguistic competence through inference rules, thereby separating innate linguistic knowledge from the contingently acquired lexicon, our approach unifies generative grammar and reinforcement learning, hence potentially resolving the still pending Chomsky-Skinner controversy.

Peter beim Graben
Fraunhofer Institute for Ceramic Technologies and Systems IKTS
Project Group "Cognitive Material Diagnostics"
Brandenburgische Technische Universität (BTU) Cottbus–Senftenberg
Platz der Deutschen Einheit 1
D – 03046 Cottbus, e-mail: `peter.beimgraben@b-tu.de`

Ronald Römer · Werner Meyer · Markus Huber · Matthias Wolff
Department of Communication Engineering
Brandenburgische Technische Universität (BTU) Cottbus–Senftenberg
Platz der Deutschen Einheit 1
D – 03046 Cottbus

1 Introduction

Speech-controlled user interfaces [64] such as Amazon's *Alexa*, Apple's *Siri* or *Cortana* by Microsoft substantially facilitate the operation of devices and household functions to laymen. Instead of using keyboard and display as input-output interfaces, the operator pronounces requests or instructions to the device and listens to its responses. One important future development will be Smart Home and Ambient Assisted Living applications in the health sciences [36].

State-of-the-art language technology scans the acoustically analyzed speech signal for relevant keywords that are subsequently inserted into semantic frames [17, 39] to interpret the user's intent. This *slot-filling* procedure [4, 37, 59] is based on large language corpora that are evaluated by standard machine learning methods, such as conditional random fields [59] or by deep neural networks [37], for instance. The necessity to overcome traditional slot-filling by proper cognitive information and communication technologies [6] has already been emphasized by Allan [3]. His research group trains semantic parsers from large language data bases such as WordNet or VerbNet that are constrained by hand-crafted expert knowledge and semantic ontologies [2, 4, 5, 43].

One particular demand on cognitive user interfaces are the processing and understanding of declarative or imperative sentences. Consider, e.g., a speech-controlled heating device, a *cognitive heating* [15, 29, 58, 62, 63], with the operator's utterance "I am cold!" This declarative sentence must firstly be analyzed syntactically to assigning "I" to the subject position and attributing "am cold" to the predicate. Secondly, a semantic analysis interprets "I" as the speaker and "am cold" as a particular subjective state. From this representation of the speaker's state, the system must compute logical inferences and then respond accordingly, by increasing the room temperature and probably by giving a linguistic feedback signal "I increase the temperature to 22 degrees". Technically, this could be achieved using feature-value relations (FVR) [62] as semantic representations and modified Markov-decision processes (MDP) for behavior control [58, 62, 63].

Recent research in computational linguistics has demonstrated that quite different grammar formalisms, such as tree-adjoining grammar [26], multiple context-free grammar [46], range concatenation grammar [9], and minimalist grammar [48, 52] converge toward universal description models [27, 33, 38, 49]. Minimalist grammar (MG) has been developed by Stabler [48] to mathematically codify Chomsky's *Minimalist Program* [11] in the generative grammar framework. A minimalist grammar consists of a mental lexicon storing linguistic signs as arrays of syntactic, phonetic and semantic features, on the one

hand, and of two structure-building functions, called "merge" and "move", on the other hand. Syntactic features in the lexicon are, e.g., the linguistic base types noun (n), verb (v), adjective (a), determiner (d), inflection (i.e. tense t), or, preposition (p). These are syntactic heads selecting other categories either as complements or as adjuncts. The structure generation is controlled by selector categories that are "merged" together with their selected counterparts. Moreover, one distinguishes between licensors and licensees, triggering the movement of maximal projections. An MG does not comprise any phrase structure rules; all syntactic information is encoded in the feature array of the mental lexicon. Furthermore, syntax and compositional semantics can be combined via the lambda calculus [30, 42], while MG parsing can be straightforwardly implemented through bottom-up [24], top-down [24, 35, 50], and in the meantime also by left-corner automata [53].

One important property of MG is their effective learnability in the sense of Gold's formal learning theory [19]. Specifically, MG can be acquired by positive examples [7, 31, 51] from linguistic dependence graphs [8, 28, 41], which is consistent with psycholinguistic findings on early-child language acquisition [14, 16, 18, 44, 57]. However, learning through positive examples only, could easily lead to overgeneralization. According to Pinker [44] this could substantially be avoided through reinforcement learning [47, 56]. Although there is only little psycholinguistic evidence for reinforcement learning in human language acquisition [40, 55], we outline a machine learning algorithm for the acquisition of an MG mental lexicon [21] of the syntax and semantics for English declarative sentences through reinforcement learning in this chapter. Instead of looking at pure syntactic dependencies as in [7, 31, 51], our approach directly uses their underlying semantic dependencies for the simultaneous segmentation of syntax and semantics.

2 Minimalist Grammar

Our language acquisition approach for minimalist grammar combines methods from computational linguistics, formal logic, and abstract algebra. Starting point of our algorithm are *utterance meaning pairs* (UMP) [21, 61].

$$u = \langle e, \sigma \rangle, \tag{1}$$

where $e \in E$ is the spoken or written utterance, given as the *exponent* of a linguistic sign [32]. Technically, exponents are strings taken from the Kleene hull of some finite alphabet, A, i.e. $E = A^*$. The sign's *semantics* $\sigma \in \Sigma$ is a logical term, expressed by means of predicate logic and the (untyped) lambda calculus [12].

2.1 Semantics

As an example, we consider the simple UMP

$$u = \langle \texttt{the mouse eats cheese}, \texttt{eat(cheese)(mouse)} \rangle \qquad (2)$$

in the sequel. We use typewriter font throughout the chapter to emphasize that utterances are regarded plainly as symbolic tokens without any intended meaning in the first place. This applies even to the "semantic" representation in terms of first order predicate logic where we use the Schönfinkel-Curry [34, 45] notation here. Therefore, the expression above $\texttt{eat(cheese)(mouse)}$ indicates that $\texttt{eat}$ is a binary predicate, fetching first its direct object $\texttt{cheese}$ to form a unary predicate, $\texttt{eat(cheese)}$, that then takes its subject $\texttt{mouse}$ in the second step to build the proposition of the utterance (2).

Following Kracht [32], we regard a linguistic sign as an ordered triple

$$z = \langle e, t, \sigma \rangle \qquad (3)$$

with the same exponent $e \in E$ and semantics $\sigma \in \Sigma$ as in the UMP (1). In addition, $t \in T$ is a syntactic *type* that we encode by means of minimalist grammar (MG) in its chain representation [52]. The type controls the generation of syntactic structure and hence the order of lambda application, analogously to the typed lambda calculus [13] in Montague semantics [34]. In order to avoid redundancy, we use the plain (untyped) lambda calculus in the following [12].

Lambda calculus is a mathematical formalism developed by Church in the 1930s "to model the mathematical notion of substitution of values for bound variables" according to [60]. Although the original application was in the area of computability (cf. [12]) the substitution of parts of a term with other terms is often the central notion when lambda calculus is used. This is also true in our case and we have to clarify the concepts first. Namely *variable*, *bound* and *free*, *term*, and *substitution*.

To be applicable to any universe of discourse one prerequisite of lambda calculus is that variables are elements from "an enumerably infinite set of symbols" [12]. However, for usage in a specific domain, a finite set is sufficient. Since we aim at terms from first order predicate logic, treating them with the operations from lambda calculus, all their predicates P and individuals I need to be in the set of variables. Additionally, we will use the symbols $I_V := \{\texttt{x}, \texttt{y}, \ldots\}$ as variables for individuals and $T_V := \{\texttt{P}, \texttt{Q}, \ldots\}$ as variables for (parts of) logical terms. The set $V := P \cup I \cup I_V \cup T_V$ is thus used as the set of variables. Note, that the distinction made by I_V and T_V is not on the level of lambda calculus but rather a visual clue to the reader.

The term algebra of lambda calculus is inductively defined as follows. *i*) Every variable $v \in V$ is a term and v is a *free* variable in the term v; specifically, also every well-formed formula of predicate logic is a term. *ii*) Given a term T and a variable $v \in V$ which is free in T, the expression $\lambda v.T$ is also a term and the variable v is now *bound* in $\lambda v.T$. Every other variable in T different from v is free (bound) in $\lambda v.T$ if it is free (bound) in T. *iii*) Given two terms T and U, the expression $T(U)$ is also a term and every variable which is free (bound) in T or U is free (bound) in $T(U)$. Such a term is often referred to as *operator-operand combination* [60] or *functional application* [34]. For disambiguation we also allow parentheses around terms. The introduced syntax differs from the original one where additionally braces and brackets are used to mark the different types of terms (cf. [12]). Sometimes, $T(U)$ is also written as (TU) and the dot between λ and the variable is left out (cf. [60]).

For a given variable $v \in V$ and two terms T and U the operation of *substitution* is $T[v \leftarrow U]$ (originally written as $S_U^v T \,|$ in [12] and sometimes without the right bar, i.e. as in [60]) and stands for the result of substituting U for all instances of v in T.

Church defined three conversions based on substitution.

- *Renaming* bound variables by replacing any part $\lambda v.T$ of a term by $\lambda w.T[v \leftarrow w]$ when the variable w does not occur in the term T.

- *Lambda application* by replacing any part $\lambda v.T(U)$ of a term by $T[v \leftarrow U]$, when the bound variables in T are distinct both from v and the free variables in U.

- *Lambda abstraction* by replacing any part $T[v \leftarrow U]$ of a term by $\lambda v.T(U)$, when the bound variables in T are distinct both from v and the free variables in U.

The first conversion simply states that names of bound variables have no particular meaning on their own. The second and third conversions are of special interest to our aims. Lambda application allows the composition of logical terms out of predicates, individuals and other logical terms while lambda abstraction allows the creation of templates of logical terms.

Applied to our example (2), we have the sign

$$z = \langle \texttt{the mouse eats cheese}, \texttt{:c}, \texttt{eat(cheese)(mouse)} \rangle \qquad (4)$$

where the now appearing MG type $\texttt{:c}$ indicates that the sign is complex (not lexical) and a complementizer phrase of type $\texttt{c}$. Its compositional se-

mantics [34] can be described by the terms $\lambda\texttt{P}.\lambda\texttt{Q}.\texttt{P}(\texttt{Q})$ and $\lambda\texttt{P}.\lambda\texttt{Q}.\texttt{Q}(\texttt{P})$, the predicate $\texttt{eat}$ and the individuals $\texttt{cheese}$ and $\texttt{mouse}$. Consider the term $\lambda\texttt{P}.\lambda\texttt{Q}.\texttt{P}(\texttt{Q})(\texttt{eat})(\texttt{cheese})$. This is converted by two successive lambda applications via $\lambda\texttt{Q}.\texttt{eat}(\texttt{Q})(\texttt{cheese})$ into the logical term $\texttt{eat}(\texttt{cheese})$. It is also possible to rearrange parts of the term in a different way. Consider now the term $\lambda\texttt{P}.\lambda\texttt{Q}.\texttt{Q}(\texttt{P})$, the logical term $\texttt{eat}(\texttt{cheese})$ and the individual $\texttt{mouse}$. Then the term $\lambda\texttt{P}.\lambda\texttt{Q}.\texttt{Q}(\texttt{P})(\texttt{mouse})(\texttt{eat}(\texttt{cheese}))$ is converted by two successive lambda applications into the logical term $\texttt{eat}(\texttt{cheese})(\texttt{mouse})$. Thus, logical terms can be composed through lambda application.

Moreover, given the logical term $\texttt{eat}(\texttt{cheese})(\texttt{mouse})$ two successive lambda abstractions yield the term $\lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat}(\texttt{x})(\texttt{y})$, leaving out the operand parts. In that way, templates of logical terms are created where different individuals can be inserted for term evaluation. Both processes are crucial for our utterance-meaning transducer and machine language acquisition algorithms below.

2.2 Syntax

An MG consists of a data base, the mental lexicon, containing signs as arrays of syntactic, phonetic and semantic *features*, and of two structure-generating functions, called "merge" and "move". Syntactic features are the *basic types* $b \in B$ from a finite set B, with $b = \texttt{n}, \texttt{v}, \texttt{a}, \texttt{d},$ etc. together with a set of their respective *selectors* $S = \{\texttt{=}b | b \in B\}$ that are unified by the "merge" operation. Moreover, one distinguishes between a set of licensers $L_+ = \{\texttt{+}l | l \in L\}$ and another set of their corresponding licensees $L_- = \{\texttt{-}l | l \in L\}$ triggering the "move" operation. L is another finite set of movement identifiers. $F = B \cup S \cup L_+ \cup L_-$ is called the feature set. Finally, one has a two-element set $C = \{::, :\}$ of categories,[1] where "::" indicates *simple, lexical* categories while ":" denotes *complex, derived* categories. The ordering of syntactic features as they appear originally in the lexicon is prescribed as regular expressions, i.e. $T = C(S \cup L_+)^* B L_-^*$ is the set of syntactic lexical types [48, 52]. The set of linguistic signs is then given as $Z = E \times T \times \Sigma$ [32].

Let $e_1, e_2 \in E$ be exponents, $\sigma_1, \sigma_2 \in \Sigma$ semantic terms in the lambda calculus, $f \in B \cup L$ one feature identifier, $\mathbf{t}, \mathbf{t}_1, \mathbf{t}_2 \in F^+$ feature strings compatible with the regular types in T, $\cdot \in C$ and $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2 \in Z^*$ sequences of signs, then $\langle e_1, ::= f\mathbf{t}_1, \sigma_1 \rangle$ and $\langle e_2, : f, \sigma_2 \rangle$ form signs in the sense of (3).

---

[1] Departing from the convention in the literature [52], we call the elements of C *categories* due to a tentative interpretation in terms of indexed grammars [1, 54]. We shall address this interesting issue in subsequent research.

A sequence of signs is called a *minimalist expression*, and the first sign of an expression is called its *head*, controlling the structure building through "merge" and "move" as follows.

The MG function "merge" is defined through inference schemata

$$\frac{\langle e_1, ::=f\mathbf{t}, \sigma_1\rangle \quad \langle e_2, \cdot f, \sigma_2\rangle \mathbf{z}}{\langle e_1 e_2, :\mathbf{t}, \sigma_1\sigma_2\rangle \mathbf{z}} \text{ merge-1}, \tag{5}$$

$$\frac{\langle e_1, :=f\mathbf{t}, \sigma_1\rangle \mathbf{z}_1 \quad \langle e_2, \cdot f, \sigma_2\rangle \mathbf{z}_2}{\langle e_2 e_1, :\mathbf{t}, \sigma_1\sigma_2\rangle \mathbf{z}_1\mathbf{z}_2} \text{ merge-2}, \tag{6}$$

$$\frac{\langle e_1, \cdot=f\mathbf{t}_1, \sigma_1\rangle \mathbf{z}_1 \quad \langle e_2, \cdot f\mathbf{t}_2, \sigma_2\rangle \mathbf{z}_2}{\langle e_1, :\mathbf{t}_1, \sigma_1\rangle \mathbf{z}_1\langle e_2, :\mathbf{t}_2, \sigma_2\rangle \mathbf{z}_2} \text{ merge-3}, \tag{7}$$

Correspondingly, "move" is given through

$$\frac{\langle e_1, :+f\mathbf{t}, \sigma_1\rangle \mathbf{z}_1\langle e_2, :-f, \sigma_2\rangle \mathbf{z}_2}{\langle e_2 e_1, :\mathbf{t}, \sigma_1\sigma_2\rangle \mathbf{z}_1\mathbf{z}_2} \text{ move-1}, \tag{8}$$

$$\frac{\langle e_1, :+f\mathbf{t}_1, \sigma_1\rangle \mathbf{z}_1\langle e_2, :-f\mathbf{t}_2, \sigma_2\rangle \mathbf{z}_2}{\langle e_1, :\mathbf{t}_1, \sigma_1\rangle \mathbf{z}_1\langle e_2, :\mathbf{t}_2, \sigma_2\rangle \mathbf{z}_2} \text{ move-2}. \tag{9}$$

where only one sign with licensee $-f$ may appear in the expression licensed by $+f$ in the head. This so-called *shortest movement constraint* (SMC) guarantees syntactic locality demands [48, 52].

A minimalist derivation terminates when all syntactic features besides one distinguished *start symbol*, which is c in our case, have been consumed. We conventionally use complementizer phrase c as the start symbol.

For illustrating the rules $(5-9)$ and their applicability, let us stick with the example UMP (2). Its syntactic analysis in terms of generative grammar [22] yields the (simplified) phrase structure tree in Fig. 1(a).[2]

---

[2] For the sake of simplicity we refrain from presenting full-fledged X-bar hierarchies [22].

(a)

```
                CP
              /    \
           DP        IP
          /  \      /   \
         D    N    I      I′
         |    |    |     /  \
       the  mouse eat_i  I    VP
                         |   /  \
                        -s  V    N
                            |    |
                          ε_i  cheese
```

(b)

```
              eat(cheese)(mouse)
                  /        \
      λy.eat(cheese)(y)    mouse
          /          \
  λx.λy.eat(x)(y)    cheese
```
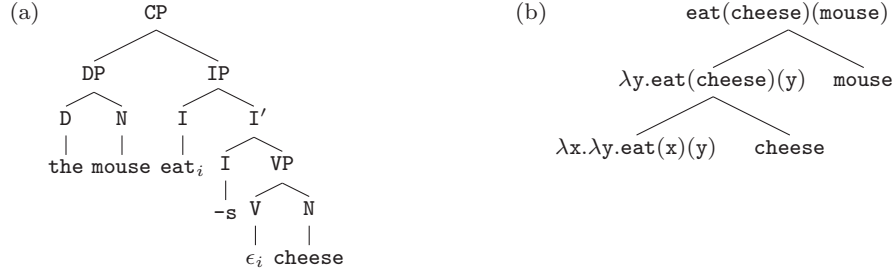
Fig. 1   Generative grammar analysis of example UMP (2). (a) Syntactic phrase structure tree. (b) Semantic tree from lambda calculus.

The syntactic categories in Fig. 1(a) are the *maximal projections* CP (complementizer phrase), IP (inflection phrase), VP (verbal phrase), and DP (determiner phrase). Furthermore, there are the intermediary node I′ and the *heads* I (inflection), D (determiner), V (verb), and N (noun), corresponding to t, d, v, and n in MG, respectively. Note that inflection is lexically realized only by the present tense suffix -s. Moreover, the verb eat has been moved out of its base-generated position leaving the empty string ϵ there. Movement is indicated by co-indexing with i.

Correspondingly, we present a simple semantic analysis in Fig. 1(b) using the notation from Sect. 2.1 together with the lambda calculus of the binary predicate in its Schönfinkel-Curry representation [34, 45].

Guided by the linguistic analyses in Fig. 1, an expert could construe a minimalist lexicon as given in Tab. 1 by hand [52].

Table 1   Minimalist lexicon for example grammar Fig. 1.

$$\langle \text{mouse}, ::\text{n}, \text{mouse} \rangle \qquad \langle \text{cheese}, ::\text{n } -\text{k}, \text{cheese} \rangle$$
$$\langle \text{the}, ::=\text{n d } -\text{k}, \epsilon \rangle \qquad \langle \text{eat}, ::=\text{n v } -\text{f}, \lambda\text{x}.\lambda\text{y}.\text{eat(x)(y)} \rangle$$
$$\langle -\text{s}, ::=\text{pred } +\text{f } +\text{k t}, \epsilon \rangle \ \langle \epsilon, ::=\text{v } +\text{k } =\text{d pred}, \lambda\text{P}.\lambda\text{Q}.\text{Q(P)} \rangle$$
$$\langle \epsilon, ::=\text{t c}, \epsilon \rangle$$

We adopt a shallow semantic model, where the universe of discourse only contains two individuals, the mouse and a piece of cheese.[3] Then, the lexicon Tab. 1 is interpreted as follows. Since all entries are contained in the MG lexicon, they are of category "::". There are two nouns (n), mouse and cheese

---

[3]   Moreover, we abstract our analysis from temporal and numeral semantics and also from the intricacies of the semantics of noun phrases in the present exposition.

with their respective semantics as individual constants, `mouse` and `cheese`. In contrast to `mouse`, the latter possesses a licensee `-k` for case marking. The same holds for the determiner `the` selecting a noun (`=n`) as its complement to form a determiner phrase `d` which also requires case assignment (`-k`) afterwards. The verb (`v`) `eat` selects a noun as a complement and has to be moved for inflection `-f`. Its compositional semantics is given by the binary predicate `eat(x)(y)` whose argument variables are bounded by two lambda expressions $\lambda x.\lambda y$. Moreover, we have an inflection suffix `-s` for present tense in third person singular, taking a predicate (`pred`) as complement, then triggering firstly inflection movement `+f` and secondly case assignment `+k`, whose type is tense (`t`). Finally, there are two entries that are phonetically not realized. The first one selects a verbal phrase `=v` and assigns case `+k` afterwards; then, it selects a determiner phrase `=d` as subject and has its own type predicate `pred`; additionally, we prescribe an intertwiner of two abstract lambda expressions `Q, P` as its semantics. The last entry provides a simple type conversion from tense `t` to complementizer `c` in order to arrive at a well-formed sentence with start symbol `c`.

Using the lexicon Tab. 1, the sign (4) is obtained by the minimalist derivation (10).

$$\frac{\langle\texttt{the, ::=n d -k}, \epsilon\rangle \qquad \langle\texttt{mouse, ::n}, \texttt{mouse}\rangle}{\langle\texttt{the mouse, :d -k}, \texttt{mouse}\rangle}\ \text{merge-1} \tag{10-1}$$

$$\frac{\langle\texttt{eat, ::=n v -f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle \qquad \langle\texttt{cheese, ::n -k}, \texttt{cheese}\rangle}{\langle\texttt{eat, :v -f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{cheese, :-k}, \texttt{cheese}\rangle}\ \text{merge-3} \tag{10-2}$$

$$\frac{\langle\epsilon, \texttt{::=v +k =d pred}, \lambda P.\lambda Q.Q(P)\rangle \qquad \langle\texttt{eat, :v -f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{cheese, :-k}, \texttt{cheese}\rangle}{\langle\epsilon, \texttt{:+k =d pred}, \lambda P.\lambda Q.Q(P)\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{cheese, :-k}, \texttt{cheese}\rangle}\ \text{merge-3} \tag{10-3}$$

$$\frac{\langle\epsilon, \texttt{:+k =d pred}, \lambda P.\lambda Q.Q(P)\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{cheese, :-k}, \texttt{cheese}\rangle}{\langle\texttt{cheese, :=d pred}, (\lambda P.\lambda Q.Q(P))(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle}\ \text{move-1} \tag{10-4}$$

$$\frac{\langle\texttt{cheese, :=d pred}, (\lambda P.\lambda Q.Q(P))(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle}{\langle\texttt{cheese, :=d pred}, \lambda Q.Q(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle}\ \lambda\text{-app.} \tag{10-5}$$

$$\frac{\langle\texttt{cheese, :=d pred}, \lambda Q.Q(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle \qquad \langle\texttt{the mouse, :d -k}, \texttt{mouse}\rangle}{\langle\texttt{cheese, :pred}, \lambda Q.Q(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}\ \text{merge-3} \tag{10-6}$$

$$\frac{\langle\texttt{-s, ::=pred +f +k t}, \epsilon\rangle \qquad \langle\texttt{cheese, :pred}, \lambda Q.Q(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}{\langle\texttt{-s cheese, :+f +k t}, \lambda Q.Q(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}\ \text{merge-1} \tag{10-7}$$

$$\frac{\langle\texttt{-s cheese, :+f +k t}, \lambda Q.Q(\texttt{cheese})\rangle\langle\texttt{eat, :-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}{\langle\texttt{eat-s cheese, :+k t}, (\lambda Q.Q(\texttt{cheese}))(\lambda x.\lambda y.\texttt{eat(x)(y)})\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}\ \text{move-1} \tag{10-8}$$

$$\frac{\langle\texttt{eats cheese, :+k t}, (\lambda Q.Q(\texttt{cheese}))(\lambda x.\lambda y.\texttt{eat(x)(y)})\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}{\langle\texttt{eats cheese, :+k t}, (\lambda x.\lambda y.\texttt{eat(x)(y)})(\texttt{cheese})\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}\ \lambda\text{-app.} \tag{10-9}$$

$$\frac{\langle\texttt{eats cheese, :+k t}, (\lambda x.\lambda y.\texttt{eat(x)(y)})(\texttt{cheese})\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}{\langle\texttt{eats cheese, :+k t}, \lambda y.\texttt{eat(cheese)(y)}\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}\ \lambda\text{-app.} \tag{10-10}$$

$$\frac{\langle\texttt{eats cheese, :+k t}, \lambda y.\texttt{eat(cheese)(y)}\rangle\langle\texttt{the mouse, :-k}, \texttt{mouse}\rangle}{\langle\texttt{the mouse eats cheese, :t}, (\lambda y.\texttt{eat(cheese)(y)})(\texttt{mouse})\rangle}\ \text{move-1} \tag{10-11}$$

$$\frac{\langle\texttt{the mouse eats cheese, : t}, (\lambda y.\texttt{eat(cheese)(y)})(\texttt{mouse})\rangle}{\langle\texttt{the mouse eats cheese, :t}, \texttt{eat(cheese)(mouse)}\rangle}\ \lambda\text{-app.} \tag{10-12}$$

$$\frac{\langle\epsilon, \texttt{::=t c}, \epsilon\rangle \qquad \langle\texttt{the mouse eats cheese, :t}, \texttt{eat(cheese)(mouse)}\rangle}{\langle\texttt{the mouse eats cheese, :c}, \texttt{eat(cheese)(mouse)}\rangle}\ \text{merge-1}\ . \tag{10-13}$$

In the first step, (10-1), the determiner the takes the noun mouse as its complement to form a determiner phrase d that requires licensing through case marking afterwards. In step 2, the finite verb eat selects the noun cheese as direct object, thus forming a verbal phrase v. As there remain unchecked features, only merge-3 applies yielding a minimalist expression, i.e. a sequence of signs. In step 3, the phonetically empty predicate pred merges with the formerly built verbal phrase. Since pred assigns accusative case, the direct object is moved in (10-4) toward the first position through case marking by simultaneously concatenating the respective lambda terms. Thus, lambda application entails the expression in step 5. Then, in step 6, the predicate selects its subject, the formerly construed determiner phrase. In the seventh step, (10-7), the present-tense suffix unifies with the predicate, entailing an inflection phrase pred, whose verb is moved into the first position in step 8, thereby yielding the inflected verb eat-s. In steps 9 and 10 two lambda applications result into the correct semantics, already displayed in Fig. 1(b). Step 11 assigns nominative case to the subject through movement into spec-

ifier position. A further lambda application in step 12 yields the intended interpretation of predicate logics. Finally, in step 13, the syntactic type `t` is converted into `c` to obtain the proper start symbol of the grammar.

2.3 Utterance-Meaning Transducer

Derivations such as (10) are essential for minimalist grammar. However, their computation is neither incremental nor predictive. Therefore, they are not suitable for natural language processing in their present form of data-driven bottom-up processing. A number of different parsing architectures have been suggested in the literature to remedy this problem [24, 35, 50, 53]. From a psycholinguistic point of view, predictive parsing appears most plausible, because a cognitive agent should be able to make informed guesses about a speaker's intents as early as possible, without waiting for the end of an utterance [23]. This makes either an hypothesis-driven top-down parser, or a mixed-strategy left-corner parser desirable also for language engineering applications. In this section, we briefly describe a bidirectional utterance-meaning transducer (UMT) for MG that is based upon Stabler's top-down recognizer [50] as outlined earlier in [20]. Its generalization towards the recent left-corner parser [53] is straightforward.

The central object for MG language processing is the *derivation tree* obtained from a bottom-up derivation as in (10). Figure 2 depicts this derivation tree, where we present a comma-separated sequence of exponents for the sake of simplicity. Additionally, every node is addressed by an index tuple that is computed according to Stabler's algorithm [50].
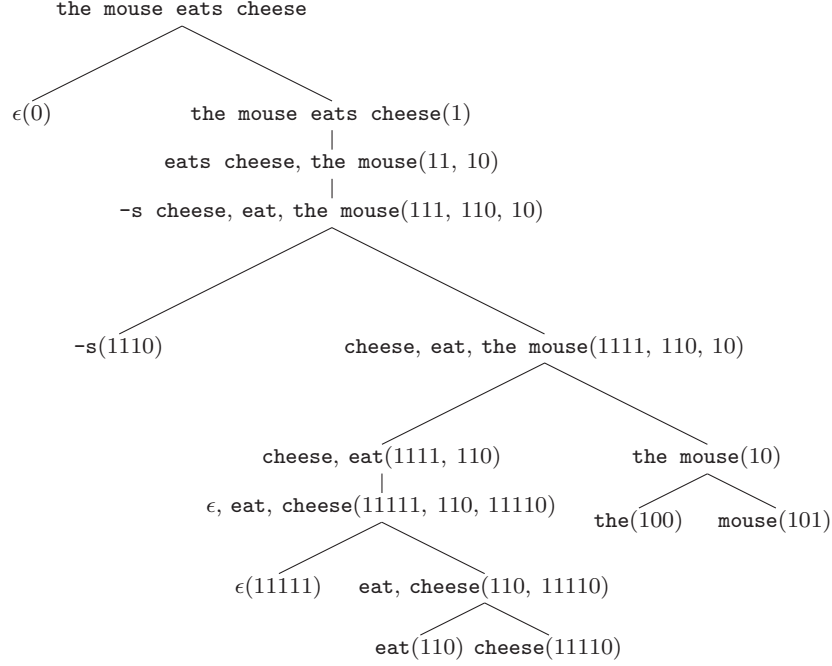
```
                    the mouse eats cheese
                       ╱          ╲
      ε(0)                    the mouse eats cheese(1)
                                        │
                          eats cheese, the mouse(11, 10)
                                        │
                    -s cheese, eat, the mouse(111, 110, 10)
                              ╱                    ╲
          -s(1110)                    cheese, eat, the mouse(1111, 110, 10)
                                           ╱                      ╲
                        cheese, eat(1111, 110)            the mouse(10)
                                   │                        ╱        ╲
                   ε, eat, cheese(11111, 110, 11110)   the(100)   mouse(101)
                            ╱            ╲
                ε(11111)      eat, cheese(110, 11110)
                                  ╱        ╲
                           eat(110)    cheese(11110)
```

Fig. 2 Simplified derivation tree of (10). Exponents of different signs are separated by commas. Nodes are also addressed by index tuples.

Pursuing the tree paths in Fig. 2 from the bottom to the top, provides exactly the derivation (10). However, reading it from the top towards the bottom allows for an interpretation in terms of *multiple context-free grammars* [38,46] (MCFG) where categories are n-ary predicates over string exponents. Like in context-free grammars, every branching in the derivation tree Fig. 2 leads to one phrase structure rule in the MCFG. Thus, the MCFG enumerated in (11) codifies the MG Tab. 1.

$$\langle:c\rangle(e_0 e_1) \leftarrow \langle::=t\ c\rangle(e_0) \quad \langle:t\rangle(e_1) \tag{11-1}$$

$$\langle:t\rangle(e_1 e_0) \leftarrow \langle:+k\ t,\ -k\rangle(e_0, e_1) \tag{11-2}$$

$$\langle:+k\ t,\ -k\rangle(e_1 e_0, e_2) \leftarrow \langle:+f\ +k\ t,\ -f,\ -k\rangle(e_0, e_1, e_2) \tag{11-3}$$

$$\langle:+f\ +k\ t,\ -f,\ -k\rangle(e_0 e_1, e_2, e_3) \leftarrow \langle::=pred\ +f\ +k\ t\rangle(e_0) \quad \langle:pred,\ -f,\ -k\rangle(e_1, e_2, e_3) \tag{11-4}$$

$$\langle:pred,\ -f,\ -k\rangle(e_0, e_1, e_2) \leftarrow \langle:=d\ pred,\ -f\rangle(e_0, e_1) \quad \langle:d\ -k\rangle(e_2) \tag{11-5}$$

$$\langle:=d\ pred,\ -f\rangle(e_2 e_0, e_1) \leftarrow \langle:+k\ =d\ pred,\ -f,\ -k\rangle(e_0, e_1, e_2) \tag{11-6}$$

$$\langle:+k\ =d\ pred,\ -f,\ -k\rangle(e_0, e_1, e_2) \leftarrow \langle::=v\ +k\ =d\ pred\rangle(e_0) \quad \langle:v\ -f,\ -k\rangle(e_1, e_2) \tag{11-7}$$

$$\langle:v\ -f,\ -k\rangle(e_0, e_1) \leftarrow \langle::=n\ v\ -f\rangle(e_0) \quad \langle::n\ -k\rangle(e_1) \tag{11-8}$$

$$\langle:d\ -k\rangle(e_0 e_1) \leftarrow \langle::=n\ d\ -k\rangle(e_0) \quad \langle::n\rangle(e_1) \tag{11-9}$$

$$\langle::n\rangle(\texttt{mouse}) \tag{11-10}$$

$$\langle::n\ -k\rangle(\texttt{cheese}) \tag{11-11}$$

$$\langle::=n\ d\ -k\rangle(\texttt{the}) \tag{11-12}$$

$$\langle::=n\ v\ -f\rangle(\texttt{eat}) \tag{11-13}$$

$$\langle::=pred\ +f\ +k\ t\rangle(\texttt{-s}) \tag{11-14}$$

$$\langle::=v\ +k\ =d\ pred\rangle(\epsilon) \tag{11-15}$$

$$\langle::=t\ c\rangle(\epsilon) \tag{11-16}$$

In (11), the angular brackets enclose the MCFG categories that are obviously formed by tuples of MG categories and syntactic types. These categories have the same number of string arguments e_k as prescribed in the type tuples. Because MCFG serve only for syntactic parsing in our UMT, we deliberately omit the semantic terms here; they are reintroduced below. The MCFG rules (11-1 – 11-9) are directly obtained from the derivation tree Fig. 2 by reverting the merge and move operations of (10) through their "unmerge" and "unmove" counterparts [24]. The MCFG axioms, i.e. the lexical rules (11-10 – 11-16), are reformulations of the entries in the MG lexicon Tab. 1.

2.3.1 Language Production

The UMT's language production module finds a semantic representation of an intended utterance in form of a Schönfinkel-Curry [34,45] formula of predicate logic, such as $\texttt{eat}(\texttt{cheese})(\texttt{mouse})$, for instance. According to Fig. 1(b) this is a hierarchical data structure that can control the MG derivation (10). Thus, the cognitive agent accesses its mental lexicon, either through Tab. 1 or its MCFG twin (11) in order to retrieve the linguistic signs for the denotations $\texttt{eat}$, $\texttt{cheese}$, and $\texttt{mouse}$. Then, the semantic tree Fig. 1(b) governs the correct derivation (10) up to lexicon entries that are phonetically empty. These must occasionally be queried from the data base whenever required. At the end of the derivation the computed exponent $\texttt{the mouse eats cheese}$ is uttered.

2.3.2 Language Understanding

The language understanding module of our UMT comprises three memory tapes: the input sequence, a syntactic priority queue, and also a semantic priority queue. Input tape and syntactic priority queue together constitute Stabler's priority queue top-down parser [50]. Yet, in order to compute the meaning of an utterance in the semantic priority queue, we slightly depart from the original proposal by omitting the simplifying trim function. Table 2 presents the temporal evolution of the top-down recognizer's configurations while processing the utterance `the mouse eats cheese`.

Table 2   MG top-down parse of `the mouse eats cheese`.

step input	syntactic queue	operation
1. the mouse eats cheese	$\langle$:c$\rangle(\epsilon)$	expand (11-1)
2. the mouse eats cheese	$\langle$::=t c$\rangle(0)\langle$:t$\rangle(1)$	scan (11-16)
3. the mouse eats cheese	$\langle$:t$\rangle(1)$	expand (11-2)
4. the mouse eats cheese	$\langle$:+k t, -k$\rangle(11, 10)$	expand (11-3)
5. the mouse eats cheese	$\langle$:+f +k t, -f, -k$\rangle(111, 110, 10)$	expand (11-4)
6. the mouse eats cheese	$\langle$::=pred +f +k t$\rangle(1110)\langle$:pred, -f, -k$\rangle(1111, 110, 10)$	sort
7. the mouse eats cheese	$\langle$:pred, -f, -k$\rangle(1111, 110, 10)\langle$::=pred +f +k t$\rangle(1110)$	expand (11-5)
8. the mouse eats cheese	$\langle$:=d pred, -f$\rangle(1111, 110)\langle$:d -k$\rangle(10)\langle$::=pred +f +k t$\rangle(1110)$	sort
9. the mouse eats cheese	$\langle$:d -k$\rangle(10)\langle$:=d pred, -f$\rangle(1111, 110)\langle$::=pred +f +k t$\rangle(1110)$	expand (11-9)
10. the mouse eats cheese	$\langle$::=n d -k$\rangle(100)\langle$::n$\rangle(101)\langle$:=d pred, -f$\rangle(1111, 110)\langle$::=pred +f +k t$\rangle(1110)$	scan (11-12)
11. mouse eats cheese	$\langle$::n$\rangle(101)\langle$:=d pred, -f$\rangle(1111, 110)\langle$::=pred +f +k t$\rangle(1110)$	scan (11-10)
12. eats cheese	$\langle$:=d pred, -f$\rangle(1111, 110)\langle$::=pred +f +k t$\rangle(1110)$	expand (11-6)
13. eats cheese	$\langle$:+k =d pred, -f, -k$\rangle(11111, 110, 11110)\langle$::=pred +f +k t$\rangle(1110)$	expand (11-7)
14. eats cheese	$\langle$::=v +k =d pred$\rangle(11111)\langle$:v -f, -k$\rangle(110, 11110)\langle$::=pred +f +k t$\rangle(1110)$	sort
15. eats cheese	$\langle$:v -f, -k$\rangle(110, 11110)\langle$::=pred +f +k t$\rangle(1110)\langle$::=v +k =d pred$\rangle(11111)$	expand (11-8)
16. eats cheese	$\langle$::=n v -f$\rangle(110)\langle$::n -k$\rangle(11110)\langle$::=pred +f +k t$\rangle(1110)\langle$::=v +k =d pred$\rangle(11111)$	sort
17. eats cheese	$\langle$::=n v -f$\rangle(110)\langle$::=pred +f +k t$\rangle(1110)\langle$::n -k$\rangle(11110)\langle$::=v +k =d pred$\rangle(11111)$	scan (11-13)
18. -s cheese	$\langle$::=pred +f +k t$\rangle(1110)\langle$::n -k$\rangle(11110)\langle$::=v +k =d pred$\rangle(11111)$	scan (11-14)
19. cheese	$\langle$::n -k$\rangle(11110)\langle$::=v +k =d pred$\rangle(11111)$	scan (11-11)
20. ϵ	$\langle$::=v +k =d pred$\rangle(11111)$	scan (11-15)
21. ϵ	ϵ	accept

The parser is initialized with the input string to be processed and the MCFG start symbol $\langle$:c$\rangle(\epsilon)$ — corresponding to the MG start symbol `c` — at the top of the priority queue. For each rule of the MCFG (11), the algorithm replaces its string variables by an index tuple that addresses the corresponding nodes in the derivation tree Fig. 2 [50]. These indices allow for an ordering relation where shorter indices are smaller than longer ones, while indices of equal length are ordered lexicographically. As a consequence, the MCFG axioms in (11) become ordered according to their temporal appearance in the utterance. Using the notation of the derivation tree Fig. 2, we get

$$\texttt{the}(100) < \texttt{mouse}(101) < \texttt{eat}(110) < \texttt{-s}(1110) < \texttt{cheese}(11110) \,.$$

Hence, index sorting ensures incremental parsing.

Besides the occasional sorting of the syntactic priority queue, the automaton behaves as a conventional context-free top-down parser. When the first item in the queue is an MCFG category appearing at the left hand side of an MCFG rule, this item is *expand*ed into the right hand side of that rule. When the first item in the queue is a predicted MCFG axiom whose exponent also appears on top of the input tape, this item is *scan*ned from the input and thereby removed from queue and input simultaneously. Finally, if queue and input both contain only the empty word, the utterance has been successfully recognized and the parser terminates in the *accept*ing state.

Interestingly, the index formalism leads to a straightforward implementation of the UMT's semantic parser as well. The derivation tree Fig. 2 reveals that the index length correlates with the tree depth. In our example, the items $\langle$::n -k$\rangle$(11110) and $\langle$::=v +k =d pred$\rangle$(11111) in the priority queue have the longest indices. These correspond precisely to the lambda terms cheese and λP.λQ.Q(P), respectively, that are unified by lambda application in derivation step (10-5). Moreover, also the semantic analysis in Fig. 1(b) illustrates that the deepest nodes are semantically unified first.

Every time, when the syntactic parser scans a correctly predicted item from the input tape, this item is removed from both input tape and syntactic priority queue. Simultaneously, the semantic content of its sign is pushed on top of the semantic priority queue, yet preserving its index. When some or all semantic items are stored in the queue, they are sorted in *reversed* index order to get highest semantic priority on top of the queue. Table 3 illustrates the semantic parsing for the given example.

Table 3 Semantic processing of the mouse eats cheese.

step input	semantic queue	operation
1. the mouse eats cheese ϵ		scan (11-16)
2. the mouse eats cheese	$\langle\epsilon\rangle(0)$	apply
3. the mouse eats cheese ϵ		scan (11-12)
4. mouse eats cheese	$\langle\epsilon\rangle(100)$	apply
5. mouse eats cheese	ϵ	scan (11-10)
6. eats cheese	$\langle$mouse$\rangle(101)$	scan (11-13)
7. -s cheese	$\langle$mouse$\rangle(101)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)$	scan (11-14)
8. cheese	$\langle$mouse$\rangle(101)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)\langle\epsilon\rangle(1110)$	apply
9. cheese	$\langle$mouse$\rangle(101)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)$	scan (11-11)
10. ϵ	$\langle$mouse$\rangle(101)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)\langle$cheese$\rangle(11110)$	scan (11-15)
11. ϵ	$\langle$mouse$\rangle(101)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)\langle$cheese$\rangle(11110)\langle\lambda$P.$\lambda$Q.Q(P)$\rangle(11111)$	sort
11. ϵ	$\langle\lambda$P.λQ.Q(P)$\rangle(11111)\langle$cheese$\rangle(11110)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)\langle$mouse$\rangle(101)$	apply
12. ϵ	$\langle\lambda$Q.Q(cheese)$\rangle(1111)\langle\lambda$x.$\lambda$y.eat(x)(y)$\rangle(110)\langle$mouse$\rangle(101)$	apply
13. ϵ	$\langle(\lambda$x.λy.eat(x)(y))(cheese)$\rangle(111)\langle$mouse$\rangle(101)$	apply
14. ϵ	$\langle\lambda$y.eat(cheese)(y)$\rangle(111)\langle$mouse$\rangle(101)$	apply
15. ϵ	$\langle$eat(cheese)(mouse)$\rangle(11)$	understand

In analogy to the syntactic recognizer, the semantic parser operates in similar modes. Both processors share their common *scan* operation. In contrast

to the syntactic parser which sorts indices in ascending order, the semantic module *sort*s them in descending order for operating on the deepest nodes in the derivation tree first. Most of the time, it attempts lambda application (*apply*) which is always preferred for ϵ-items on the queue. When apply has been sufficiently performed to a term, the last index number is removed from its index (sometimes it might also be necessary to exchange two items for lambda application). Finally, the semantic parser terminates in the *under-stand*ing state.

3 Reinforcement Learning

Sofar we discussed how a cognitive agent, being either human or an intelligent machine, could produce and understand utterances that are described in terms of minimalist grammar. An MG is given by a mental lexicon as in example Tab. 1, encoding a large amount of linguistic expert knowledge. Therefore, it seems unlikely that speech-controlled user interfaces could be build and sold by engineering companies for little expenses.

Yet, it has been shown that MG are effectively learnable in the sense of Gold's formal learning theory [19]. The studies [7, 31, 51] demonstrated how MG can be acquired by positive examples from linguistic dependence graphs [8, 41]. The required dependency structures can be extracted from linguistic corpora by means of big data machine learning techniques, such as the expectation maximization (EM) algorithm [28].

In our terminology, such statistical learning methods only consider correlations at the exponent level of linguistic signs. By contrast, in the present study we propose an alternative training algorithm that simultaneously analyzes similarities between exponents and semantic terms. Moreover, we exploit both positive and negative examples to obtain a better performance through reinforcement learning [47, 56].

The language learner is a cognitive agent $\mathcal{L}$ in a state X_t, to be identified with $\mathcal{L}$'s mental lexicon at training time t. At time $t = 0$, $\mathcal{L}$ is initialized as a *tabula rasa* with empty lexicon

$$X_0 \leftarrow \emptyset \tag{12}$$

and exposed to UMPs produced by a teacher $\mathcal{T}$. Note that we assume $\mathcal{T}$ presenting already complete UMPs and not singular utterances to $\mathcal{L}$. Thus we circumvent the *symbol grounding problem* of firstly assigning meanings σ to uttered exponents e [25], which will be addressed in future research. Moreover,

we assume that $\mathcal{L}$ is instructed to reproduce $\mathcal{T}$'s utterances based on its own semantic understanding. This provides a feedback loop and therefore applicability of reinforcement learning [47,56]. For our introductory example, we adopt the simple semantic model from Sect. 2. In each iteration, the teacher utters an UMP that should be learned by the learner.

First iteration

Let the teacher $\mathcal{T}$ make the first utterance (2)

$$u_1 = \langle \mathtt{the\ mouse\ eats\ cheese}, \mathtt{eat(cheese)(mouse)} \rangle \, .$$

As long as $\mathcal{L}$ is not able to detect patterns or common similarities in $\mathcal{T}$'s UMPs, it simply adds new entries directly to its mental lexicon, assuming that all utterances are complex ":" and possessing base type $\mathtt{c}$, i.e. the MG start symbol. Hence, $\mathcal{L}$'s state X_t evolves according to the update rule

$$X_t \leftarrow X_{t-1} \cup \left\{ \langle e_t, \mathtt{:c}, \sigma_t \rangle \right\} \, , \tag{13}$$

when $u_t = \langle e_t, \sigma_t \rangle$ is the UMP presented at time t by $\mathcal{T}$.

In this way, the mental lexicon X_1 shown in Tab. 4 has been acquired at time $t = 1$.

Table 4 Learned minimalist lexicon X_1 at time $t = 1$.

$$\langle \mathtt{the\ mouse\ eats\ cheese}, \mathtt{:c}, \mathtt{eat(cheese)(mouse)} \rangle$$

Second iteration

Next, let the teacher utter another proposition

$$u_2 = \langle \mathtt{the\ rat\ eats\ cheese}, \mathtt{eat(cheese)(rat)} \rangle \, . \tag{14}$$

Looking at u_1, u_2 together, the agent's pattern matching module is able to find similarities between exponents and semantics, underlined in (15).

$$u_1 = \langle \underline{\text{the}} \text{ mouse } \underline{\text{eats cheese}}, \underline{\text{eat}(\text{cheese})}(\text{mouse}) \rangle \qquad (15\text{-}1)$$

$$u_2 = \langle \underline{\text{the}} \text{ rat } \underline{\text{eats cheese}}, \underline{\text{eat}(\text{cheese})}(\text{rat}) \rangle . \qquad (15\text{-}2)$$

Thus, $\mathcal{L}$ creates two distinct items for `the mouse` and `the rat`, respectively, and carries out lambda abstraction to obtain the updated lexicon X_2 in Tab. 5.

Table 5 Learned minimalist lexicon X_2 at time $t = 2$.

$$\langle \underline{\text{the}} \text{ mouse}, \text{:d}, \text{mouse} \rangle \qquad\qquad \langle \underline{\text{the}} \text{ rat}, \text{:d}, \text{rat} \rangle$$
$$\langle \text{eats cheese}, \text{:=d c}, \lambda\text{y.eat}(\text{cheese})(\text{y}) \rangle$$

Note that the induced variable symbol `y` and syntactic types `d`, `c` are completely arbitrary and do not have any particular meaning to the agent.

As indicated by underlines in Tab. 5, the exponents `the mouse` and `the rat`, could be further segmented through pattern matching, that is not reflected by their semantic counterparts, though. Therefore, a revised lexicon X_{21}, displayed in Tab. 6 can be devised.

Table 6 Revised minimalist lexicon X_{21}.

$$\langle \text{the}, \text{::=n d}, \epsilon \rangle \ \langle \text{mouse}, \text{::n}, \text{mouse} \rangle$$
$$\langle \text{rat}, \text{::n}, \text{rat} \rangle \ \ \langle \text{eats cheese}, \text{:=d c}, \lambda\text{y.eat}(\text{cheese})(\text{y}) \rangle$$

For closing the reinforcement cycle, $\mathcal{L}$ is supposed to produce utterances upon its own understanding. Thus, we assume that $\mathcal{L}$ wants to express the proposition `eat(cheese)(rat)`. According to our discussion in Sect. 2.3.1, the corresponding signs are retrieved from the lexicon X_{21} and processed through a valid derivation leading to the correct utterance `the rat eats cheese`, that is subsequently endorsed by $\mathcal{T}$.

Third iteration

In the third training session, the teacher's utterance might be

$$u_3 = \langle \text{the mouse eats carrot}, \text{eat}(\text{carrot})(\text{mouse}) \rangle . \qquad (16)$$

Now we have to compare u_3 with the lexicon entry for `eats cheese` in (17).

$$\langle \text{the mouse \underline{eats} carrot}, \underline{\text{eat}}(\text{carrot})(\text{mouse})\rangle \qquad (17\text{-}1)$$

$$\langle \underline{\text{eats}} \text{ cheese}, \text{:c}, \lambda\text{y}.\underline{\text{eat}}(\text{cheese})(\text{y})\rangle . \qquad (17\text{-}2)$$

Another lambda abstraction entails the lexicon X_3 in Tab. 7.

Table 7 Learned minimalist lexicon X_3 at time $t = 3$.

$\langle \text{the}, ::\text{=n d}, \epsilon\rangle$ $\qquad \langle \text{mouse}, ::\text{n}, \text{mouse}\rangle$
$\langle \text{rat}, ::\text{n}, \text{rat}\rangle$ $\qquad \langle \text{cheese}, ::\text{n}, \text{cheese}\rangle$
$\langle \text{carrot}, ::\text{n}, \text{carrot}\rangle$ $\langle \text{eats}, ::\text{=n =d c}, \lambda\text{x}.\lambda\text{y}.\text{eat}(\text{x})(\text{y})\rangle$

Here, the learner assumes that `eats` is a simple, lexical category without having further evidence as in Sect. 2.

Since $\mathcal{L}$ is instructed to produce well-formed utterances, it could now generate a novel semantic representation, such as `eat(carrot)(rat)`. This leads through data base query from the mental lexicon X_3 to the correct derivation (18) that is rewarded by $\mathcal{T}$.

$$\frac{\langle \text{the}, ::\text{=n d}, \epsilon\rangle \qquad \langle \text{rat}, ::\text{n}, \text{rat}\rangle}{\langle \text{the rat}, :\text{d}, \text{rat}\rangle} \text{merge-1} \qquad (18\text{-}1)$$

$$\frac{\langle \text{eats}, ::\text{=n =d c}, \lambda\text{x}.\lambda\text{y}.\text{eat}(\text{x})(\text{y})\rangle \qquad \langle \text{carrot}, ::\text{n}, \text{carrot}\rangle}{\langle \text{eats carrot}, :\text{=d c}, (\lambda\text{x}.\lambda\text{y}.\text{eat}(\text{x})(\text{y}))(\text{carrot})\rangle} \text{merge-1} \qquad (18\text{-}2)$$

$$\frac{\langle \text{eats carrot}, :\text{=d c}, (\lambda\text{x}.\lambda\text{y}.\text{eat}(\text{x})(\text{y}))(\text{carrot})\rangle}{\langle \text{eats carrot}, :\text{=d c}, \lambda\text{y}.\text{eat}(\text{carrot})(\text{y})\rangle} \lambda\text{-appl.} \qquad (18\text{-}3)$$

$$\frac{\langle \text{eats carrot}, :\text{=d c}, \lambda\text{y}.\text{eat}(\text{carrot})(\text{y})\rangle \qquad \langle \text{the rat}, :\text{d}, \text{rat}\rangle}{\langle \text{the rat eats carrot}, :\text{c}, (\lambda\text{y}.\text{eat}(\text{carrot})(\text{y}))(\text{rat})\rangle} \text{merge-2} \qquad (18\text{-}4)$$

$$\frac{\langle \text{the rat eats carrot}, :\text{c}, (\lambda\text{y}.\text{eat}(\text{carrot})(\text{y}))(\text{rat})\rangle}{\langle \text{the rat eats carrot}, :\text{c}, \text{eat}(\text{carrot})(\text{rat})\rangle} \lambda\text{-appl.} \qquad (18\text{-}5)$$

Forth iteration

In the fourth iteration, we suppose that $\mathcal{T}$ utters

$$u_4 = \langle \text{the rats eat cheese}, \text{eat}(\text{cheese})(\text{rats})\rangle \qquad (19)$$

that is unified with the previous lexicon X_3 through our pattern matching algorithm to yield X_4 in Tab. 8 in the first place.

Table 8   Learned minimalist lexicon X_4 at time $t = 4$.

$\langle\texttt{the}, ::=\texttt{n d}, \epsilon\rangle$ $\qquad\qquad$ $\langle\texttt{mouse}, ::\texttt{n}, \texttt{mouse}\rangle$
$\langle\underline{\texttt{rat}}, ::\texttt{n}, \underline{\texttt{rat}}\rangle$ $\qquad\qquad\quad$ $\langle\underline{\texttt{rat}}\texttt{s}, ::\texttt{n}, \underline{\texttt{rat}}\texttt{s}\rangle$
$\langle\texttt{cheese}, ::\texttt{n}, \texttt{cheese}\rangle$ $\qquad\quad$ $\langle\texttt{carrot}, ::\texttt{n}, \texttt{carrot}\rangle$
$\langle\underline{\texttt{eat}}\texttt{s}, ::=\texttt{n =d c}, \lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)}\rangle$ $\langle\underline{\texttt{eat}}, ::=\texttt{n =d c}, \lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)}\rangle$

Underlined are again common strings in exponents or semantics that could entail further revisions of the MG lexicon.

Next, let us assume that $\mathcal{L}$ would express the meaning $\texttt{eat(carrot)(rats)}$. It could then attempt the following derivation (20).

$$\frac{\langle\texttt{the}, ::=\texttt{n d}, \epsilon\rangle \qquad \langle\texttt{rats}, ::\texttt{n}, \texttt{rats}\rangle}{\langle\texttt{the rats}, :\texttt{d}, \texttt{rats}\rangle} \text{ merge-1} \qquad\qquad (20\text{-}1)$$

$$\frac{\langle\texttt{eats}, ::=\texttt{n =d c}, \lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)}\rangle \qquad \langle\texttt{carrot}, ::\texttt{n}, \texttt{carrot}\rangle}{\langle\texttt{eats carrot}, :=\texttt{d c}, (\lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)})(\texttt{carrot})\rangle} \text{ merge-1} \qquad (20\text{-}2)$$

$$\frac{\langle\texttt{eats carrot}, :=\texttt{d c}, (\lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)})(\texttt{carrot})\rangle}{\langle\texttt{eats carrot}, :=\texttt{d c}, \lambda\texttt{y}.\texttt{eat(carrot)(y)}\rangle} \text{ λ-appl.} \qquad\qquad (20\text{-}3)$$

$$\frac{\langle\texttt{eats carrot}, :=\texttt{d c}, \lambda\texttt{y}.\texttt{eat(carrot)(y)}\rangle \qquad \langle\texttt{the rats}, :\texttt{d}, \texttt{rats}\rangle}{\langle\texttt{the rats eats carrot}, :\texttt{c}, (\lambda\texttt{y}.\texttt{eat(carrot)(y)})(\texttt{rats})\rangle} \text{ merge-2} \qquad (20\text{-}4)$$

$$\frac{\langle\texttt{the rats eats carrot}, :\texttt{c}, (\lambda\texttt{y}.\texttt{eat(carrot)(y)})(\texttt{rats})\rangle}{\langle\texttt{the rats eats carrot}, :\texttt{c}, \texttt{eat(carrot)(rats)}\rangle} \text{ λ-appl.} \qquad\qquad (20\text{-}5)$$

However, uttering `the rats eats carrot` will probably be rejected by the teacher $\mathcal{T}$ because of the grammatical number agreement error, thus causing punishment by $\mathcal{T}$. As a consequence, $\mathcal{L}$ has to find a suitable revision of its lexicon X_4 that is guided by the underlined matches in Tab. 8.

To this aim, the agent first modifies X_4 as given in Tab. 9.

Table 9   Revised minimalist lexicon X_{41}.

$\langle\texttt{the}, ::=\texttt{num d}, \epsilon\rangle$ $\qquad\qquad$ $\langle\texttt{mouse}, ::\texttt{n -a}, \texttt{mouse}\rangle$
$\langle\texttt{rat}, ::\texttt{n -a}, \texttt{rat}\rangle$ $\qquad\qquad$ $\langle\epsilon, ::=\texttt{n +a num}, \epsilon\rangle$
$\langle\texttt{-s}, ::=\texttt{n +a num}, \epsilon\rangle$ $\qquad\quad$ $\langle\texttt{cheese}, ::\texttt{n}, \texttt{cheese}\rangle$
$\langle\texttt{carrot}, ::\texttt{n}, \texttt{carrot}\rangle$ $\qquad\quad$ $\langle\underline{\texttt{eat}}\texttt{s}, ::=\texttt{n =d c}, \lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)}\rangle$
$\langle\underline{\texttt{eat}}, ::=\texttt{n =d c}, \lambda\texttt{x}.\lambda\texttt{y}.\texttt{eat(x)(y)}\rangle$

In Tab. 9 the entries for `mouse` and `rat` have been updated by a number licensee `-a` (for *Anzahl*). Moreover, the entry for `the` now selects a number type `=num` instead of a noun. Even more crucially, two novel entries of number type `num` have been added: a phonetically empty item $\langle \epsilon, \texttt{::=n +a num}, \epsilon \rangle$ selecting a noun `=n` and licensing number movement `+a`, and an item for the plural suffix $\langle \texttt{-s}, \texttt{::=n +a num}, \epsilon \rangle$ with the same feature sequence.

Upon the latter revision, the agent may successfully derive `rat`, `rats`, and `mouse`, but also `mouses`, which will be rejected by the teacher. In order to avoid punishment, the learner had to wait for the well-formed item `mice` once to be uttered by $\mathcal{T}$. Yet, the current evidence prevents the agent from correctly segmenting `eats`, because our shallow semantic model does not sufficiently constrain any further pattern matching. This could possibly be remedied in case of sophisticated numeral and temporal semantic models. At the end of the day, we would expect something alike the hand-crafted lexicon Tab. 1 from Sect. 2. For now, however, we leave this important problem for future research.

4 Discussion

In this contribution we have outlined an algorithm for effectively learning the syntax and semantics of English declarative sentences. Such sentences are presented to a cognitive agent by a teacher in form of utterance meaning pairs (UMP) where the meanings are encoded as formulas of first order predicate logic. This representation allows for the application of compositional semantics via lambda calculus [12]. For the description of syntactic categories we use Stabler's minimalist grammar [48, 52], (MG) a powerful computational implementation of Chomsky's recent Minimalist Program for generative linguistics [11]. Despite the controversy between Chomsky and Skinner [10], we exploit reinforcement learning [47, 56] as training paradigm. Since MG codifies universal linguistic competence through the five inference rules (5 – 9), thereby separating innate linguistic knowledge from the contingently acquired lexicon, our approach could potentially unify generative grammar and reinforcement learning, hence resolving the abovementioned dispute.

Minimalist grammar can be learned from linguistic dependency structures [8, 28, 31, 51] by positive examples, which is supported by psycholinguistic findings on early human language acquisition [16, 44, 57]. However, as Pinker [44] has emphasized, learning through positive examples alone, could lead to undesired overgeneralization. Therefore, reinforcement learning that might play a role in children language acquisition as well [40, 55], could effec-

tively avoid such problems. The required dependency structures are directly provided by the semantics in the training UMPs. Thus, our approach is explicitly semantically driven, in contrast to the algorithm in [28] that regards dependencies as latent variables for EM training.

As a proof-of-concept we suggested an algorithm for simple English declarative sentences. We also have evidence that it works for German and French as well and hopefully for other languages also. Our approach will open up an entirely new avenue for the further development of speech-controlled cognitive user interfaces [6, 58, 64].

References

1. Aho, A.V.: Indexed grammars — an extension of context-free grammars. Journal of the Association for Computing Machinery **15**(4), 647 − 671 (1968).
2. Allen, J.: Learning a lexicon for broad-coverage semantic parsing. In: Proceedings of the ACL 2014 Workshop on Semantic Parsing, pp. 1 − 6 (2014).
3. Allen, J.: Dialogue as collaborative problem solving. In: Proceedings of Interspeech Conference, p. 833 (2017)
4. Allen, J.F.: Natural language processing. In: Encyclopedia of Computer Science, pp. 1218 − 1222. Wiley, Chichester (UK) (2003)
5. Allen, J.F., Bahkshandeh, O., de Beaumont, W., Galescu, L., Teng, C.M.: Effective broad-coverage deep parsing. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (2018).
6. Baranyi, P., Csapo, A., Sallai, G.: Cognitive Infocommunications (CogInfoCom). Springer (2015).
7. Bonato, R., Retoré, C.: Learning rigid Lambek grammars and minimalist grammars from structured sentences. In: Proceedings of the Third Workshop on Learning Language in Logic, pp. 23 − 34 (2001).
8. Boston, M., Hale, J., Kuhlmann, M.: Dependency structures derived from minimalist grammars. In: C. Ebert, G. Jäger, J. Michaelis (eds.) The Mathematics of Language, *Lecture Notes in Computer Science*, vol. 6149, pp. 1 − 12. Springer, Berlin (2010).
9. Boullier, P.: Range concatenation grammars. In: H. Bunt, J. Carroll, G. Satta (eds.) New Developments in Parsing Technology, *Text, Speech and Language Technology*, vol. 23, pp. 269 − 289. Springer (2005).
10. Chomsky, N.: A review of b. f. skinner's *verbal behavior*. Language **35**(1), 26 − 58 (1959).
11. Chomsky, N.: The Minimalist Program. No. 28 in Current Studies in Linguistics. MIT Press, Cambridge (MA) (1995).
12. Church, A.: An unsolvable problem of elementary number theory. American Journal of Mathematics **58**(2), 345 − 363 (1936).
13. Church, A.: A formulation of the simple theory of types. Journal of Symbolic Logic **5**(2), 56 − 68 (1940).
14. Diessel, H.: Construction grammar and first language acquisition. In: G. Trousdale, T. Hoffmann (eds.) The Oxford Handbook of Construction Grammar, pp. 347 − 364. Oxford University Press, Oxford (2013).

15. Duckhorn, F., Huber, M., Meyer, W., Jokisch, O., Tschöpe, C., Wolff, M.: Towards an autarkic embedded cognitive user interface. In: F. Lacerda (ed.) Proceedings of the Interspeech Conference Stockholm, pp. 3435 – 3436 (2017)

16. Ellis, N.C.: Language acquisition as rational contingency learning. Applied Linguistics **27**(1), 1 – 24 (2006).

17. Fillmore, C.J.: The mechanisms of "construction grammar". In: Proceedings of the Annual Meeting of the Berkeley Linguistics Society, vol. 14 (1988)

18. Gee, J.P.: First language acquisition as a guide for theories of learning and pedagogy. Linguistics and Educatation **6**(4), 331 – 354 (1994).

19. Gold, E.M.: Language identification in the limit. Information and Control **10**(5), 447 – 474 (1967).

20. beim Graben, P., Meyer, W., Römer, R., Wolff, M.: Bidirektionale Utterance-Meaning-Transducer für Zahlworte durch kompositionale minimalistische Grammatiken. In: P. Birkholz, S. Stone (eds.) Tagungsband der 30. Konferenz Elektronische Sprachsignalverarbeitung (ESSV), *Studientexte zur Sprachkommunikation*, vol. 91, pp. 76 – 82. TU-Dresden Press, Dresden (2019)

21. beim Graben, P., Römer, R., Meyer, W., Huber, M., Wolff, M.: Reinforcement learning of minimalist numeral grammars. In: Proceedings of IEEE Conference on Cognitive Infocommunications, Naples, p. 13 (2019).

22. Haegeman, L.: Introduction to Goverment & Binding Theory, *Blackwell Textbooks in Linguistics*, vol. 1, 2nd edn. Blackwell Publishers, Oxford (1994).

23. Hale, J.T.: What a rational parser would do. Cognitive Science **35**(3), 399 – 443 (2011).

24. Harkema, H.: Parsing minimalist languages. Ph.D. thesis, University of California, Los Angeles (2001)

25. Harnad, S.: The symbol grounding problem. Physica D **42**, 335 – 346 (1990)

26. Joshi, A.K., Levy, L.S., Takahashi, M.: Tree adjunct grammars. Journal of Computer and System Sciences **10**(1), 136 – 163 (1975).

27. Joshi, A.K., Vijay-Shanker, K., Weir, D.: The convergence of mildly context-sensitive grammar formalisms. Tech. Rep. MS-CIS-90-01, University of Pennsylvania (1990)

28. Klein, D., Manning, C.D.: Corpus-based induction of syntactic structure: models of dependency and constituency. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg (PA) (2004).

29. Klimczak, P., Wolff, M., Lindemann, J., Petersen, C., Römer, R., Zoglauer, T.: Die kognitive Heizung. In: R. Hoffmann (ed.) Elektronische Sprachsignalverarbeitung (ESSV), vol. 25, pp. 89 – 96. TU Dresden Press, Dresden (2014)

30. Kobele, G.M.: Syntax and semantics in minimalist grammars. In: Proceedings of ESSLLI 2009 (2009)

31. Kobele, G.M., Collier, T., Taylor, C., Stabler, E.P.: Learning mirror theory. In: Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 66 – 73 (2002).

32. Kracht, M.: The Mathematics of Language. No. 63 in Studies in Generative Grammar. Mouton de Gruyter, Berlin (2003)

33. Kuhlmann, M., Koller, A., Satta, G.: Lexicalization and generative power in CCG. Computational Linguistics **41**(2), 215 – 247 (2015).

34. Lohnstein, H.: Formale Semantik und natrliche Sprache. De Gryuter, Berlin (2011)

35. Mainguy, T.: A probabilistic top-down parser for minimalist grammars. ArXiv cs.CL 1010.1826 (2010).

36. Martin, M., Weibel, R., Röcke, C., Boker, S.M.: Semantic activity analytics for healthy aging: challenges and opportunities. IEEE Pervasive Computing **17**(3), 73 – 77 (2018).

37. Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., Zweig, G.: Using recurrent neural networks for slot filling in spoken language understanding. IEEE Transactions on Audio, Speech and Language Processing **23**(3), 530 – 539 (2015).

38. Michaelis, J.: Derivational minimalism is mildly context-sensitive. In: M. Moortgat (ed.) Logical Aspects of Computational Linguistics, *Lecture Notes in Artificial Intelligence*, vol. 2014, pp. 179 – 198. Springer, Berlin (2001)

39. Minsky, M.: A framework for representing knowledge. Tech. Rep. AIM-306, M.I.T., Cambridge (MA) (1974).

40. Moerk, E.L.: A behavioral analysis of controversial topics in first language acquisition: Reinforcements, corrections, modeling, input frequencies, and the three-term contingency pattern. Journal of Psycholinguistic Research **12**(2), 129 – 155 (1983).

41. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of the 8th International Workshop on Parsing Technologies (IWPT, pp. 149 – 160 (2003).

42. Niyogi, S.: A minimalist implementation of verb subcategorization. In: Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001). (2001)

43. Perera, I., Allen, J., Teng, C.M., Galescu, L.: Building and learning structures in a situated blocks world through deep language understanding. In: Proceedings of the First International Workshop on Spatial Language Understanding, pp. 12 – 20 (2018).

44. Pinker, S.: Language acquisition. In: L.R. Gleitman, D.N. Osherson, M. Liberman, L.R. Gleitman, D.N. Osherson, M. Liberman (eds.) Language: An Invitation to Cognitive Science, chap. 6, pp. 135 – 182. MIT Press (1995)

45. Schönfinkel, M.: Über die Bausteine der mathematischen Logik. Mathematische Annalen **92**(3), 305 – 316 (1924).

46. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science **88**(2), 191 – 229 (1991).

47. Skinner, B.F.: Verbal Behavior. Martino Publishing, Mansfield Centre (CT) (2015). 1st Edition 1957

48. Stabler, E.P.: Derivational minimalism. In: C. Retoré (ed.) Logical Aspects of Computational Linguistics, *Lecture Notes in Computer Science*, vol. 1328, pp. 68 – 95. Springer, New York (1997).

49. Stabler, E.P.: Computational perspectives on minimalism. In: C. Boeckx (ed.) Oxford Handbook of Linguistic Minimalism, pp. 617 – 641. Oxford University Press (2011)

50. Stabler, E.P.: Top-down recognizers for MCFGs and MGs. In: Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics, pp. 39 – 48. Association for Computational Linguistics, Portland, Oregon, USA (2011).

51. Stabler, E.P., Collier, T.C., Kobele, G.M., Lee, Y., Lin, Y., Riggle, J., Yao, Y., Taylor, C.E.: The learning and emergence of mildly context sensitive languages. In: W.B. et al. (ed.) Advances in Artificial Life, *Lecture Notes in Computer Science*, vol. 2801, pp. 525 – 534. Springer, Berlin (2003)

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

53. Stanojević, M., Stabler, E.: A sound and complete left-corner parsing for minimalist grammars. In: M. Idiart, A. Lenci, T. Poibeau, A. Villavicencio (eds.) Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing, pp. 65 – 74. Association for Computational Linguistics (ACL) (2018).

54. Staudacher, P.: New frontiers beyond context-freeness: DI-grammars and DI-automata. In: Proceedings of the Sixth Conference on European Chapter of the

Association for Computational Linguistics (EACL'93), pp. 358 – 367. Association for Computational Linguistics, Stroudsburg (PA) (1993).

55. Sundberg, M.L., Michael, J., Partington, J.W., Sundberg, C.A.: The role of automatic reinforcement in early language acquisition. Analysis of Verbal Behavior **13**(1), 21 – 37 (1996).

56. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press (2018)

57. Tomasello, M.: First steps toward a usage-based theory of language acquisition. Cognitive Linguistics **11**(1-2), 61 (2006).

58. Tschöpe, C., Duckhorn, F., Huber, M., Meyer, W., Wolff, M.: A cognitive user interface for a multi-modal human-machine interaction. In: A. Karpov, O. Jokisch, R. Potapova (eds.) Speech and Computer, pp. 707 – 717. Springer, Cham (2018)

59. Tur, G., Hakkani-Tür, D., Heck, L., Parthasarathy, S.: Sentence simplification for spoken language understanding. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5628 – 5631 (2011).

60. Wegner, P.: Lambda Calculus. In: A. Ralston, E.D. Reilly, D. Hemmendinger (eds.), Encyclopedia of Computer Science, pp. 953 - 955. John Wiley and Sons Ltd, GBR (2003).

61. Wirsching, G., Lorenz, R.: Towards meaning-oriented language modeling. In: Proceedings of the 4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), pp. 369 – 374 (2013).

62. Wolff, M., Meyer, W., Römer, R.: Modellierung von Bewältigungsverhalten mit Merkmal-Werte-Relationen. In: G. Wirsching (ed.) Tagungsband, 26. Konferenz Elektronische Sprachsignalverarbeitung, pp. 224 – 231 (2015)

63. Wolff, M., Römer, R., Wirsching, G.: Towards coping and imagination for cognitive agents. In: Proceedings of the 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) 2015, pp. 307 – 312 (2015).

64. Young, S.: Cognitive user interfaces. IEEE Signal Processing Magazine **27**(3), 128 –140 (2010).