frontiers in Computer Science

HYPOTHESIS AND THEORY
published: 27 January 2022
doi: 10.3389/fcomp.2022.733596

Unifying Physical Interaction, Linguistic Communication, and Language Acquisition of Cognitive Agents by Minimalist Grammars

Ronald Römer[1]*, *Peter beim Graben*[2], *Markus Huber-Liebl*[1] and *Matthias Wolff*[1]

[1] Brandenburg University of Technology Cottbus–Senftenberg, Cottbus, Germany, [2] Bernstein Center for Computational Neuroscience, Berlin, Germany

Cognitive agents that act independently and solve problems in their environment on behalf of a user are referred to as autonomous. In order to increase the degree of autonomy, advanced cognitive architectures also contain higher-level psychological modules with which needs and motives of the agent are also taken into account and with which the behavior of the agent can be controlled. Regardless of the level of autonomy, successful behavior is based on interacting with the environment and being able to communicate with other agents or users. The agent can use these skills to learn a truthful knowledge model of the environment and thus predict the consequences of its own actions. For this purpose, the symbolic information received during the interaction and communication must be converted into representational data structures so that they can be stored in the knowledge model, processed logically and retrieved from there. Here, we firstly outline a grammar-based transformation mechanism that unifies the description of physical interaction and linguistic communication and on which the language acquisition is based. Specifically, we use minimalist grammar (MG) for this aim, which is a recent computational implementation of generative linguistics. In order to develop proper cognitive information and communication technologies, we are using utterance meaning transducers (UMT) that are based on semantic parsers and a *mental lexicon*, comprising syntactic and semantic features of the language under consideration. This lexicon must be acquired by a cognitive agent during interaction with its users. To this aim we outline a reinforcement learning algorithm for the acquisition of syntax and semantics of English utterances. 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.

Keywords: cognitive agents, cognitive architectures, artificial intelligence, reinforcement learning, interaction and communication, minimalist grammars, semantic representations, utterance meaning transducer

OPEN ACCESS

Edited by:
Anna Esposito, University of Campania 'Luigi Vanvitelli, Italy

Reviewed by:
Farhan Mohamed, University of Technology Malaysia, Malaysia
Javad Khodadoust, Instituto de Tecnologa y Educacin Superior de Monterrey (ITESM), Mexico

***Correspondence:**
Ronald Römer
ronald.roemer@b-tu.de

Specialty section:
This article was submitted to Human-Media Interaction, a section of the journal Frontiers in Computer Science

Received: *30 June 2021*
Accepted: *04 January 2022*
Published: *27 January 2022*

Citation:
Römer R, beim Graben P, Huber-Liebl M and Wolff M (2022) Unifying Physical Interaction, Linguistic Communication, and Language Acquisition of Cognitive Agents by Minimalist Grammars. Front. Comput. Sci. 4:733596. doi: 10.3389/fcomp.2022.733596

1. INTRODUCTION

Traditionally, the technical replication of cognitive systems is based on cognitive architectures with which the most important principles of human cognition are captured. The best known traditional architectures are SOAR and ACT (Funke, 2006). Such architectures serve to integrate psychological findings in a formal model that is as economical as possible and capable of being simulated. It assumes that all cognitive processes can be traced back to a few basic principles. According to Eliasmith (2013), this means that cognitive architectures have suitable *representational data structures*, that they support the *composition-, adaptation- and classification principle* and that they are autonomously capable to gain knowledge by *logical reasoning and learning*. Further criteria are productivity, robustness, scalability and compactness. In psychological research, these architectures are available as computer programs, which are used to empirically test psychological theories. In contrast, the utility of cognitive architectures in artificial intelligence (AI) research lies primarily in the construction of intelligent machines and the ability to explain their behavior.

Explainability of intelligent machines is closely tied to the idea of a *physical symbol system* (PSS) (Newell and Simon, 1976). A PSS takes physical symbols from its sensory equipment, composing them into symbolic structures (expressions) and transforms them to new expressions (Vera and Simon, 1993). For our approach it is crucial that the symbols or the symbol structures, respectively, can be assigned a meaning and that the transformation of these symbol structures leads to logically processable knowledge on which problem solving is based. In order to build meaningful symbols, the agent must be embedded in a *perception-action-cycle* (PAC) and it must be taken into account that the truthfulness or veridicality of the translation of sensory information into symbolic structures is not necessarily guaranteed due to deceptions or dysfunctions (Bischof, 2009). To overcome these difficulties, more sophisticated agents are able to build a dynamic model of their environment that can be simulated. In this case, the agent has to distinguish between redundancy expectations (model-based prior knowledge) and sensory data (observations), so that information from two sources has to be processed. To achieve veridicality, both pieces of information must be combined in a suitable manner (e.g., through a Bayesian model). The structure required for this kind of information processing is depicted by the inner cognitive loop in **Figure 1** and is denoted as interaction[1]. This picture illustrates that autonomous behavior can arise if the interaction process is based on truthful information processing that is controlled by higher-level psychological modules. This includes necessities, motives and acquired authorizations that support well-being. In addition to the associated autonomous interaction scenarios, human-machine communication is another area of application for cognitive agents. In this case, the tool or service character of cognitive agents plays a prominent role. This property allows the user to solve certain tasks or problems through the agent. For this purpose, linguistic descriptions must be articulated, understood and exchanged. This results in the requirement for a speech-enabled agent and an additional outer cognitive loop (see **Figure 1**) which is denoted as communication[2].

In our application scenario, the communications between cognitive agents and human users are related to a common physical environment that is modeled through a network of objects that are in static or dynamic relationships with one another. Such objects can be described and distinguished by a set of attributes and its admissible values. Objects, attributes and relationships between objects are the information of interest to which both nonverbal interaction and verbal communication refer. However, in order to ensure veridical behavior, the representational data structures of the environment model must be compared with those that have resulted from communication and interaction. Only by using an appropriate comparison mechanism the agent can understand what is actually going on in reality and what the respective observations mean. It largely depends on this ability whether the right actions are selected to achieve the agent's goals. In case of interaction, the comparison must be made between the representational data of the sensor information and the content of the agents knowledge base. In communication, on the other hand, it is necessary that the representational data structures of the user's utterances (semantics) are referenced to the shared environmental context (Hausser, 2014). The agent can receive this information using the interaction loop. Note that this comparison may result in an adaptation of the knowledge base.

Research in computational linguistics has demonstrated that quite different grammar formalisms, such as tree-adjoining grammar (Joshi et al., 1975), multiple context-free grammar (Seki et al., 1991), range concatenation grammar (Boullier, 2005), and minimalist grammar (Stabler, 1997; Stabler and Keenan, 2003) converge toward universal description models (Joshi et al., 1990; Michaelis, 2001; Stabler, 2011a; Kuhlmann et al., 2015). Minimalist grammar has been developed by Stabler (1997) to mathematically codify Chomsky's *Minimalist Program* (Chomsky, 1995) 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. Furthermore, syntax and compositional semantics can be combined via the lambda calculus (Niyogi, 2001; Kobele, 2009), while MG parsing can be straightforwardly implemented through bottom-up (Harkema, 2001), top-down (Harkema, 2001; Mainguy, 2010; Stabler, 2011b), and in the meantime also by left-corner automata (Stanojević and Stabler, 2018).

One important property of MG is their effective learnability in the sense of Gold's formal learning theory (Gold, 1967). Specifically, MG can be acquired by positive examples (Bonato and Retoré, 2001; Kobele et al., 2002; Stabler et al., 2003) from linguistic dependence graphs (Nivre, 2003; Klein and Manning, 2004; Boston et al., 2010), which is consistent with

[1]Based on the terms used in control theory, we represent the cyclical flow of information in a circular form and denote the cyclical flow as a loop.

[2]Note, that the opposite point of view is also of interest, in which the agent learns user behavior by analyzing dialogs.

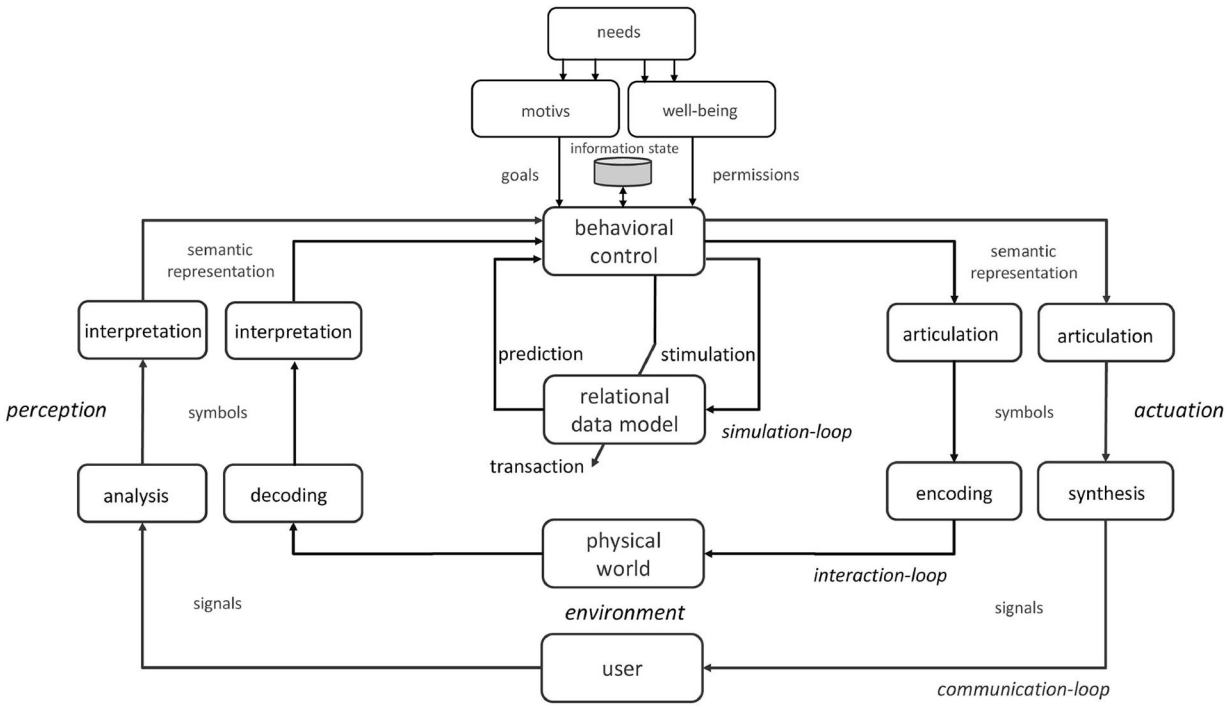

FIGURE 1 | Double cognitive loop using an embedded relational data model. The inner cognitive loop corresponds to the interaction between the agent and the environment (nonverbal information exchange). The outer cognitive loop is used for the communication between a speech-enabled agent and any other cognitive agent or natural language user (verbal information exchange). The flow of information between the behavioral control and the relational data model is embedded in a simulation loop and enables the simulation of actions as well as the prediction of effects. Since the simulation loop is not linked to the environment through perception and action, this loop is not referred to as a PAC. The exchange of information between two systems with different cognitive abilities (e.g., between humans and cognitive agents) is denoted as *Inter-Cognitive Communication* (Baranyi et al., 2015).

psycholinguistic findings on early-child language acquisition (Gee, 1994; Pinker, 1995; Ellis, 2006; Tomasello, 2006; Diessel, 2013). However, learning through positive examples only, could easily lead to overgeneralization. According to Pinker (1995) this could substantially be avoided through reinforcement learning (Skinner, 2015; Sutton and Barto, 2018). Although there is only little psycholinguistic evidence for reinforcement learning in human language acquisition (Moerk, 1983; Sundberg et al., 1996), we outline a machine learning algorithm for the acquisition of an MG mental lexicon (beim Graben et al., 2020) of the syntax and semantics for English declarative sentences through reinforcement learning in this article. Our approach chosen here is inspired by similar work on linguistic reinforcement learning (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2012; Artzi and Zettlemoyer, 2013). Instead of looking at pure syntactic dependencies as Bonato and Retoré (2001), Kobele et al. (2002), and Stabler et al. (2003), our approach directly uses their underlying semantic dependencies for the simultaneous segmentation of syntax and semantics, in some analogy to van Zaanen (2001).

With this work we are pursuing two main goals. The first goal is to represent the received or released information about the properties of objects or the relationships between objects in the form of relational expressions and to store it in a knowledge model so that these expressions can be used for knowledge processing for the purpose of behavior control or problem solving. In order to assign a meaning and a context to the symbolic structures (symbol grounding problem), the comparability between the expressions of the knowledge model and the expressions of the two cognitive loops (interaction and communication) must be guaranteed. We fulfill this requirement by uniformly describing the information transformation in both loops using the formalism of minimalist grammar and thus with linguistic means. To solve the symbol grounding problem, the relationship between the symbol structures and the actual facts is established through a model of the agent's environment. Such a model can be acquired on the principle of reinforcement learning. Since the information transformation is also based on a model—here in the form of the mental lexicon—the second aim of this work is to acquire the mental lexicon of an MG based on this learning principle.

The paper is organized as follows: First, we address the problem statement and the experimental environment. Subsequently, we summarize the necessary mathematical basics and the notations used. Then we come to the first focus of this work and describe the unified information transformation for interaction and communication using minimalist grammar and lambda calculus. We start with the simple case of interaction and explain the transformation mechanism using formal linguistic means and the assumption that the agent has already aquired a minimalist lexicon. Subsequently, the more demanding case of communication is discussed, where

the transformation mechanism has to convert natural language utterances into predicate logical expressions. In both cases we discuss language production (articulation) and language understanding (interpretation). The integration of the acquired representational data structures into the agent's knowledge base concludes the first main focus. The following section covers the second main focus and deals with the acquisition of minimalist lexicons through the method of reinforcement learning. The paper concludes with a summary and a discussion of the achieved results.

2. PROBLEM STATEMENT AND EXPERIMENTAL SETUP

We consider the well-known mouse-maze problem (Shannon, 1953; Wolff et al., 2015, 2018), where an artificial mouse lives in a simple $N \times M$ maze world that is given by a certain configuration of walls. For the mouse to survive, one ore more target objects are located at some places in the maze. In our setup we are using the target object types cheese/carrot (C) and water (W) to satisfy the primary needs hunger and thirst. These object types are defined symbolically by the set $\mathcal{O} = \{C, W, NOB\}$ where NOB refers to no object at all. The agent is able to move around the maze and to perceive information about its environment via physical interaction. This is organized by the inner perception-action-cycle (interaction-loop) of **Figure 1** that allows the agent to navigate to these target objects. To this end, the agent needs to measure its current position (x, y) by two sensors, where $x \in X = \{1, .., N\}$ and $y \in Y = \{1, .., M\}$ apply. It also has to determine the presence or absence of target objects by an object classifier, which we consider as a single complex sensor. Then the current situation can be described on the basis of the measurement result for the current position and the result of object classification. Subsequently, the measurement information needs to be encoded as a string of symbols and has to be translated into a representational data structure saved in a knowledge base. This kind of knowledge ("situations") should be stored as the result of an exploration phase if no logical contradictions occure. A further kind of knowledge is the set of movements in the maze. These "movements" are based on permissible actions $a \in \mathcal{A}$, which initially correspond to the four geographic directions, north (N), south (S), west (W), and east (E), that are defined symbolically by the set $\mathcal{A} = \{N, S, W, E, NOP\}$ (NOP denoting no operation here). Each action starts at a position $z = (x, y)$ and ends at a position $z' = (x', y')$. For this purpose, the actuators associated to the x- or y- direction, respectively, can be incremented or decremented by one step. Hence, to establish the parameterization of the four geographic directions we define the sets $\Delta X = \Delta Y = \{-1, 0, 1\}$. Thus, the south action is parameterized, for example, by the ordered pair $(0, -1)$. Note that with the knowledge about "movements" the agents behavior can be described in the sense of "causality." From a technical point of view, the relationship between a cause $[z = ((x, y), a)]$ and an effect $[z' = (x', y')]$ can be expressed, for example, by the transition equation of a finite state automaton. To implement "movement" instructions the reverse flow of the sensory information must be realized. That is, in order to be able to control the agent's actuators, actions must be described as representative data structures and converted into a string of symbols. It follows, that physical interaction requires both, the agent's capability to interpret a linearly ordered time series of symbolized measurement results as a (partially ordered) semantic representations and to transform semantic representations into a linear sequence of actuator instructions during articulation.

The ability to communicate with natural language users is another demand that a cognitive agent should meet. To this end, the agent should first of all be speech-enabled. Communication is organized by the outer perception-action-cycle (communication-loop) shown in **Figure 1** which is mainly characterized by the articulation and interpretation of speech signals. Therefore, we essentially need the very same capabilities for the transformation of linearly ordered symbolic messages (the "scores" of communication) into partially ordered semantic representations. For this reason, we devise a *bidirectional* utterance meaning transducer to encode and decode the meaning of symbolic messages. The encoded meaning corresponds to the representational data structure of utterances and can be saved in the agent's knowledge base.

In order to avoid logical conflicts between the results of the interaction- and communication loop, it is also necessary that the representational data structures generated by these loops must be comparable. Hence, in our approach non-verbal physical interaction and verbal communication are uniformly modeled using linguistic description means. This approach can be also supported by two salient arguments borrowed from Hausser (2014) that have already been presented by Römer et al. (2019): (1) "Without a carefully built physical grounding any symbolic representation will be mismatched to the sensors and actuators. These groundings provide the constraints on symbols necessary for them to be truly useful." (2) "The analysis of nonverbal cognition is needed in order to be able to plausibly explain the phylogenetic and ontogenetic development of language from earlier stages of evolution without language." Further, according to the "Physical Symbol Systems Hypothesis" (PSSH) all cognitive processes can be described as the transformation of symbol structures (Newell and Simon, 1976). This transformation process obeys the composition principle ("infinite use of finite means") and aims to recast incoming sensor information into logically processable knowledge, which is saved in a knowledge model. Notably, we assume that the transformation from signal to symbol space can be solved by a transduction stage, proposed by Wolff et al. (2013). According to the PSSH, this low-level transduction stage is the crucial processing step for recognition, designation and arrangement of the observed sensor events through symbol sequences in terms of a formal or natural language.

The use of linguistic description means includes the use of a suitable grammar formalism that is based on a mental lexicon as part of the agent's knowledge base and on transformation rules for how to arrange symbols into linear sequences. It is the second aim of the present study, to suggest minimalist grammar and logical lambda calculus as a unifying framework to this end. While the user can be assumed to already have the linguistic resources, the agent must acquire them through learning. Thus, a further aim of our work is the description of an algorithm with

which a mental lexicon can be learned and dynamically updated through reinforcement learning. Starting point of our algorithm are *utterance meaning pairs* (Kwiatkowski et al., 2012; Wirsching and Lorenz, 2013; beim Graben et al., 2019b).

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

where $e \in E$ is the spoken or written utterance while $\sigma \in \Sigma$ is a logical term, expressed by means of predicate logic and the (untyped) lambda calculus (Church, 1936) denoting the *semantics* of the utterance. We assume that UMPs are continuously delivered by a "teacher" to the agent during language acquisition.

As an example, consider the simple UMP

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

In the sequel we use typewriter font 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 (Schönfinkel, 1924; Lohnstein, 2011) notation here. Therefore, the expression above `eat(cheese)(mouse)` indicates that `eat` is a binary predicate, fetching first its direct object `cheese` to form a unary predicate, `eat(cheese)`, that then takes its subject `mouse` in the second step to build the proposition of the utterance (2).

3. PRELIMINARIES

In this section we summarize some fundamental concepts needed for system modeling and information transformation: state space representation and formal languages. We also briefly refer to the set-theoretical connection between simple predicate logic expressions (without quantifiers) and semantic representations based on relational schemes. Subsequently, we explain the concept of minimalist grammar and its suitability for a transformation mechanism that can be used to mediate between linearly ordered symbolic perception/action sequences and semi- or disordered semantic representations. The expressiveness and flexibility of natural language is mainly due to the compositional principle, according to which new semantic terms can be formed from a few elementary terms. In order to anchor this principle in the transformation mechanism, we need the approach of the lambda calculus and the representation of n-ary functions using the Schönfinkel-Curry notation.

3.1. Basic Concepts

State Space Representation. To describe the system behavior in terms of cause and effect, we distinguish states $z \in \mathcal{Z}$, actions $a \in \mathcal{A}$ and outputs[3] $o \in \mathcal{O}$. A behavioral relation is then given by

$$R_V \subseteq \mathcal{Z} \times \mathcal{A} \times \mathcal{Z} \times \mathcal{O}, \tag{3}$$

which determines the system dynamics. To decide, whether a tuple (z, a, z', o) belongs to this relation or not, the characteristic function $f_{R_V} \colon \mathcal{Z} \times \mathcal{A} \times \mathcal{Z} \times \mathcal{O} \to \{0, 1\}$ can be applied. Further, this relation corresponds to a network of causal relations. Due to causality, a recursive calculation rule divided into a system equation and an output equation is given, which allows forecasting the system's behavior

$$\begin{aligned} z' &= G(z, a) \quad \text{with} \quad G \colon \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}, \\ o &= H(z, a) \quad \text{with} \quad H \colon \mathcal{Z} \times \mathcal{A} \to \mathcal{O}. \end{aligned} \tag{4}$$

Because in our setting the system response $o \in \mathcal{O}$ depends exclusively on the current state $z \in \mathcal{Z}$, we get a simplification of the output equation $o = H(z)$ that corresponds to the idea of the Moore automaton.

Formal Language. In order to specify any technical communication between source and sink, we have to arrange an alphabet $\Sigma_A = \{a, b, \ldots\}$ and to establish a language $\mathcal{L} \subseteq {\Sigma_A}^*$ (the symbol $*$ is called Kleene star). Words or sentences are then described by an ordered sequence $\mathbf{s} = (s_1, s_2, \ldots, s_k) \in \mathcal{L}$, where $s \in \Sigma_A$. To decide whether a word s belongs to the language $\mathcal{L}$ we can devise a grammar $G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathsf{S})$. It is specified by the sets $\mathcal{N}$ of nonterminals, $\mathcal{T}$ of terminals and $\mathcal{P}$ of production rules as well as a start symbol S. The grammatical rules determine the arrangement of the symbols within a symbol sequence. Based on these rules permissible sentences of a language can be produced or derived. In contrast to sentence production, sentence analysis reveals the underlying grammatical structure. This analysis method is known as parsing and will be exploited below.

3.2. Predicate Logic Expressions

A linguistic expression that articulates the characteristics of an object or a relation between objects is called a predicate (Jungclaussen, 2001). It does not always have to be about definite objects. In the case of indefinite objects, so-called individual variables are used. Usually the letters P or Q are used as symbols for predicate names. As arguments for predicates we use either individual constants, individual variables or function values. For the sake of simplicity, we limit ourselves to individual constants and individual variables here. An n-ary predicate is then denoted by the expression $Q(a_1, a_2, \ldots, a_n)$, where a are values of the corresponding set A (e.g., admissible values from the domain of an attribute A). According to set theory the predicate $Q(a_1, a_2, \ldots, a_n)$ can be interpreted as relation $R_Q \subseteq A_1 \times A_2 \times \ldots \times A_n$ and can be understood as a binary classifier. In this case the classification task refers to finding in the set of ordered tuples of individuals or objects the class $[Q]$ of those individuals for which the predicate is true. This class is called the denotation of Q. The following notation is used for this:

$$[Q] = \{(a_1, \ldots, a_n) \in A_1 \times \ldots \times A_n \mid Q(a_1, \ldots, a_n) \text{is true}\} \tag{5}$$

The mathematical description of a predicate can also be done using the corresponding characteristic function: $f_{[Q]} : A_1 \times A_2 \times \ldots \times A_n \to \{0, 1\}$. This is particularly required for the formalism of composing semantic representations.

[3]To avoid confusion in terms of the set $\mathcal{O}$, we would like to point out that the outputs of the state space model in this work correspond to the specified object types of the mouse-maze-application.

3.3. Semantic Representations

In semantics we distinguish between objects that are described by attributes and relations between such objects (Chen, 1975). To represent semantics we will focus on feature-value pairs (FVP), which have a flat structure and correspond to tuples of database relations. The notation used here was taken from Lausen (2005). We start from a universe $\mathcal{U} = \{A_1, A_2, \ldots, A_m\}$ which comprises a finite set of attributes. Further, we have a set of domains $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ and a mapping $\mathrm{dom}: \mathcal{U} \to \mathcal{D}$. The values of the different attributes are elements of the associated domains. Based on these definitions we can specify types of objects or relations between objects by a set of attributes $\mathcal{X} = \{A_1, A_2 \ldots, A_n\} \subset \mathcal{U}$. Objects are defined as mappings, which are called tuples:

$$\tau: \{A_1, A_2, \ldots, A_n\} \to \bigcup_{i=1}^{n} \mathrm{dom}(A_i), n \leq m. \tag{6}$$

A set of such tuples corresponds to a set of distinguishable objects. Note that in database notation the elements of tuples are not ordered. This corresponds to the idea that the order of attributes is not relevant for describing the type of any object (Lausen, 2005). An object or relation type is defined by a relation scheme:

$$R(\mathcal{X}) = (A_1 : \mathrm{dom}(A_1), A_2 : \mathrm{dom}(A_2), \ldots, A_n : \mathrm{dom}(A_n)). \tag{7}$$

In database representations a scheme is associated to the head of a relational data table. The entries of a table correspond to a set of tuples. This set of tuples corresponds to a relation, which is given by:

$$R_{\mathcal{X}} \subseteq Tup(X) := \{\tau | \tau : X \to \mathrm{dom}(\mathcal{X})\}, \tag{8}$$

where $Tup(\mathcal{X})$ is the set of all possible tuples of a relation scheme $R(\mathcal{X})$. The set of all relations over this scheme is denoted as $Rel(\mathcal{X})$.

3.4. Minimalist Grammars

Minimalist grammars (MG) were introduced to describe natural languages (Stabler, 1997). For nonverbal interaction it is crucial that MG provide a translation mechanism between linearly ordered perceptions/actions and semi- or disordered semantic representations. Based on such a grammar, knowledge can be formally represented and stored consistently in a relational data model. Following Kracht (2003), we regard a linguistic sign as an ordered triple

$$q = \langle \mathsf{e}, \mathsf{t}, \sigma \rangle \tag{9}$$

with exponent $\mathsf{e} \in E$, semantics $\sigma \in \Sigma$ and a syntactic type $\mathsf{t} \in T$ that we encode by means of MG in its chain representation (Stabler and Keenan, 2003). The type controls the generation of the syntactic and semantic structure of an utterance. 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." For our purposes, it is sufficient to point out some syntactic features: Selectors (e.g., "=S") are required to search for syntactic types of the same category (e.g., "S"). Licensors (e.g., "+k") and licensees (e.g., "-k") are required to control the symbol order at the surface. The symbol "::" indicates *simple*, *lexical* categories while ":" denotes *complex*, *derived* categories ("·" is just a placeholder for one of these signs). A sequence of signs "$\mathbf{q}$" is called a *minimalist expression*. For further details we refer to beim Graben et al. (2019b). We just repeat the two kinds of structure-building rules here. The MG function "merge" is defined through inference schemes

$$\begin{aligned}
&\frac{\langle e_1, :: = f\,\mathbf{t}, \sigma_1\rangle \quad \langle e_2, \cdot f, \sigma_2\rangle \mathbf{q}}{\langle e_1 e_2, : \mathbf{t}, \sigma_1\sigma_2\rangle \mathbf{q}} && \text{merge-1}\,,\\
&\frac{\langle e_1, : = f\,\mathbf{t}, \sigma_1\rangle \mathbf{q}_1 \quad \langle e_2, \cdot f, \sigma_2\rangle \mathbf{q}_2}{\langle e_2 e_1, : \mathbf{t}, \sigma_1\sigma_2\rangle \mathbf{q}_1\mathbf{q}_2} && \text{merge-2}\,,\\
&\frac{\langle e_1, \cdot = f\,\mathbf{t}_1, \sigma_1\rangle \mathbf{q}_1 \quad \langle e_2, \cdot f\,\mathbf{t}_2, \sigma_2\rangle \mathbf{q}_2}{\langle e_1, : \mathbf{t}_1, \sigma_1\rangle \mathbf{q}_1 \langle e_2, : \mathbf{t}_2, \sigma_2\rangle \mathbf{q}_2} && \text{merge-3}\,.
\end{aligned} \tag{10}$$

Correspondingly, "move" is given through,

$$\begin{aligned}
&\frac{\langle e_1, : + f\,\mathbf{t}, \sigma_1\rangle \mathbf{q}_1 \langle e_2, : - f, \sigma_2\rangle \mathbf{q}_2}{\langle e_2 e_1, : \mathbf{t}, \sigma_1\sigma_2\rangle \mathbf{q}_1\mathbf{q}_2} && \text{move-1}\,,\\
&\frac{\langle e_1, : + f\,\mathbf{t}_1, \sigma_1\rangle \mathbf{q}_1 \langle e_2, : - f\,\mathbf{t}_2, \sigma_2\rangle \mathbf{q}_2}{\langle e_1, : \mathbf{t}_1, \sigma_1\rangle \mathbf{q}_1 \langle e_2, : \mathbf{t}_2, \sigma_2\rangle \mathbf{q}_2} && \text{move-2}\,.
\end{aligned} \tag{11}$$

3.5. Schönfinkel-Curry Notation and Lambda Calculus

Schönfinkel-Curry Notation. Let us consider a binary function

$$f: A \times B \to Z, \tag{12}$$

where its domain is defined by the cartesian product over two sets A and B and its value range is given by the set Z. Such a binary function can be decomposed into two unary functions, which are calculated in two steps

$$F: A \to (B \to Z). \tag{13}$$

First, with the assignment of the first argument $a \in A$, a mapping $F(a): B \to Z$ is selected. Secondly, the function value $F(a)(b) = f(a, b) = z$ is calculated when the second argument $b \in B$ is assigned. This principle can be generalized to n-ary functions

$$f: A_1 \times A_2 \times, \ldots, \times A_n \to A_{n+1}. \tag{14}$$

Whereby a one-to-one relationship between n-ary functions and unary functions of n-th order is established

$$F: A_1 \to (A_2 \to (A_3 \ldots (A_n \to A_{n+1}) \ldots)). \tag{15}$$

In computational linguistics this representation is used in terms of the description of relations by their characteristic function. In this case an n-tuple $(a_1, a_2, \ldots, a_n) \in A_1 \times A_2 \times, \ldots, \times A_n$ is mapped to the elements of the binary set $\{0, 1\}$. However, the representation in computational linguistics is in a slightly modified form, because the arguments appear in the reverse order $F(a_n)(a_{n-1}) \ldots (a_1)$.

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

Wegner (2003). Although the original application was in the area of computability (cf. Church, 1936) 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 a prerequisite of lambda calculus is "an enumerably infinite set of symbols" (Church, 1936) which can be used as variables. 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 $I_V := \{\mathtt{x}, \mathtt{y}, \ldots\}$ as variables for individuals and $T_V := \{\mathtt{P}, \mathtt{Q}, \ldots\}$ 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 meta-theoretical clue for the reader.

The following definitions hold for lambda calculus in general and hence we simply use "normal" typeface letters for variables. 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 resp. bound in $\lambda v.T$ if it is free resp. bound in T. *iii*) Given two terms T and U, the expression $T(U)$ is also a term and every variable which is free resp. bound in T or U is free resp. bound in $T(U)$. Such a term is often referred to as *operator-operand combination* (Wegner, 2003) or *functional application* (Lohnstein, 2011). 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. Church, 1936). Sometimes, $T(U)$ is also written as (TU) and the dot between λ and the variable is left out (cf. Wegner, 2003).

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 $S^v_U T$ in Church (1936) and sometimes without the right bar, i. e. as in Wegner (2003)] 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

$$q = \langle \texttt{the mouse eats cheese}, \texttt{:c}, \texttt{eat(cheese)(mouse)} \rangle \tag{16}$$

where the now appearing MG type `:c` indicates that the sign is complex (not lexical) and a complementizer phrase of type `c`. Its compositional semantics (Lohnstein, 2011) can be described by the terms λ`P.`λ`x.P(x)` and λ`x.`λ`P.P(x)`, the predicate `eat` and the individuals `cheese` and `mouse`. Consider the term λ`P.`λ`x.P(x)(eat)(cheese)`. This is converted by two successive lambda applications via λ`x.eat(x)(cheese)` into the logical term `eat(cheese)`. It is also possible to rearrange parts of the term in a different way. Consider now the term λ`x.`λ`P.P(x)`, the logical term `eat(cheese)` and the individual `mouse`. Then the term λ`x.`λ`P.P(x)(mouse)(eat(cheese))` is converted by two successive lambda applications into the logical term `eat(cheese)(mouse)`. Thus, logical terms can be composed through lambda application.

Moreover, given the logical term `eat(cheese)(mouse)` two successive lambda abstractions yield the term λ`x.`λ`y.eat(x)(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.

4. PHYSICAL INTERACTION

In order to explain the translation process for the interaction, we first rely on a regular grammar, which only comprises two rules: $\mathtt{S} \rightarrow \mathtt{eS}$ and $\mathtt{S} \rightarrow \mathtt{e}$ with $\mathtt{e} \in \mathcal{T}$ and $S \in \mathcal{N}$. To better understand the formalism we use the following indexed derivation scheme: $\mathtt{S}_0 \rightarrow \mathtt{e}_0\mathtt{S}_1$, $\mathtt{S}_1 \rightarrow \mathtt{e}_1\mathtt{S}_2$, $\mathtt{S}_2 \rightarrow \mathtt{e}_2\mathtt{S}_3$, $\ldots$, $\mathtt{S}_n \rightarrow \mathtt{e}_n\mathtt{S}_{n+1}$, $\mathtt{S}_{n+1} \rightarrow \mathtt{e}_{n+1}$. The inner words $\mathtt{e}_1, \mathtt{e}_2, \ldots, \mathtt{e}_n$ correspond to the values of attributes that are specified in a relational scheme $R(\mathcal{X})$. The embracing words $\mathtt{e}_0$ and $\mathtt{e}_{n+1}$ correspond to the symbols `<start>` and `<end>`, respectively, which indicate the beginning and the end of a sentence. The database semantics is made up of sets, the elements of which are assignments of values to attributes $\tau : A \rightarrow e \in \mathrm{dom}(A)$. To apply the translation formalism based on minimalist grammars we substitute semantic terms σ with $\sigma(e)$. Then we define the semantic concatenation as a set operation: $\sigma_1\sigma_2 := \sigma(e_1) \cup \sigma(e_2) := \{\sigma(e_1)\} \cup \{\sigma(e_2)\}$.

Minimalist lexicon. The MG formalism works with linguistic signs that are saved in a minimalist lexicon. In case of interaction three types of linguistic signs that are saved in a minimalist lexicon (see **Table 1**) are required.

The semantics of these sign types are represented by λ-terms using the set-theoretical union operator in the Schönfinkel-Curry-notation $\cup(\sigma(e_2))(\sigma(e_1)) := \cup(\sigma(e_1), \sigma(e_2)) = \sigma(e_1) \cup \sigma(e_2)$. According to the following formalism in each iteration step exactly one mapping is added to the semantic expression generated so far. In λ-calculus the composition is achieved

TABLE 1 | Minimalist lexicon for physical interaction.

⟨<start>, ::S_0 k, $\langle\sigma(e_0)\rangle$⟩
⟨<end>, ::=S_n +k S_{n+1}, λa . ∪ $(\sigma(e_{n+1}))$(a)⟩
⟨e_i, ::=S_{i-1} +k S_i, λa . ∪ $(\sigma(e_i))$(a)⟩

The symbols <start> = e_0 *and* <end> = e_{n+1} *encode the beginning and the end of a sequence of symbols. The embedded entries* e_i *correspond to sensory events. The index* i *with* $0 < i < n+1$ *is related to the attribute of the relational scheme. The semantics of* e_i *is given by* $\sigma(e_i)$*, whereby* $\sigma(e_0) = \sigma(e_{n+1}) = \emptyset$ *is applied.*

through consecutive λ-applications, where the position for the argument a is alternately opened and closed. In this way, mappings are added one after the other to a disordered set, which results to the entries of a relational scheme.

4.1. Interpretation

During the interpretation process, a sequence of symbols is converted into a set of mappings that describe a situation. Such situations are encoded in the form: $\mathbf{s} = (e_0, e_1, e_2, e_3, e_4)$ which corresponds in the mouse-maze scenario to messages $\mathbf{s} =$ (<start>, x, y, o, <end>). That is, the exponents contain the (x, y)-coordinates and the name of an object type $o \in \mathcal{O}$. Permissible sentences are elements of the observation language $\mathcal{L}_{obs}$. The configuration of these sentences is subject to a fixed agreement. For this purpose we assign a role to every exponent index and use a bijective mapping $\beta : \mathcal{R} \to \mathcal{X}$, which maps each role $r \in \mathcal{R}$ to the attributes $A \in \mathcal{X}$ of the relational schemes $R(\mathcal{X})$. The semantics of an exponent e_i with $0 < i < n+1$ is then defined by $\sigma(e_i) := \beta(r_i) \mapsto \omega_i \in dom(\beta(r_i))$ and $\sigma(e_0) = \sigma(e_{n+1}) = \emptyset$. Based on this role-dependent semantics we obtain with the MG formalism a linguistic structure in which the syntactic and semantic structure are built up in parallel. At its core, a *key-lock principle* is applied, in which a selector (e. g. "=S") always requests a syntactic type of the same category (e. g. "S") and licensors and licensees are used to control the order of the symbols on the surface. Note that this principle is not restricted to natural languages. Now we come to the description of the formalism using the λ-calculus. First, the symbol sequence $\mathbf{s} \in \mathcal{L}_{obs}$ is converted into a *sequence of linguistic signs* (taken from the minimalist lexicon), which the formalism processes step by step. During processing, the formalism alternates between the rules merge-3 and move-1. After processing the start type, the iterative processing of the mapping types takes place until the end type is reached. To calculate the semantics of $\mathbf{s}$, the

$$\textit{Start}: \quad \frac{\langle e_1, ::=S_0{+}kS_1\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_1))(a)\rangle \qquad \langle e_0, ::S_0\text{-}k, \sigma(e_0)\rangle}{\langle e_1, :{+}kS_1\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_1))(a)\rangle \qquad \langle e_0, :\text{-}k, \sigma(e_0)\rangle}\text{merge-3}$$

$$\frac{\langle e_1, :{+}kS_1\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_1))(a)\rangle \qquad \langle e_0, :\text{-}k, \sigma(e_0)\rangle}{\langle e_0e_1, :S_1\text{-}k, \bigcup(\sigma(e_1))(\sigma(e_0))\rangle}\text{move-1.}$$

$$\textit{Iterations}: \quad \frac{\langle e_2, ::=S_1{+}kS_2\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_2))(a)\rangle \qquad \langle e_0e_1, :S_1\text{-}k, \bigcup_{i=0}^{1}\sigma(e_i)\rangle}{\langle e_2, :{+}kS_2\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_2))(a)\rangle \qquad \langle e_0e_1, :\text{-}k, \bigcup_{i=0}^{1}\sigma(e_i)\rangle}\text{merge-3}$$

$$\frac{\langle e_2, :{+}kS_2\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_2))(a)\rangle \qquad \langle e_0e_1, :\text{-}k, \bigcup_{i=0}^{1}\sigma(e_i)\rangle}{\langle e_0e_1e_2, :S_2\text{-}k, \bigcup(\sigma(e_2))(\bigcup_{i=0}^{1}\sigma(e_i))\rangle}\text{move-1.}$$

$$\frac{\langle e_3, ::=S_2{+}kS_3\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_3))(a)\rangle \qquad \langle e_0e_1e_2, :S_2\text{-}k, \bigcup_{i=0}^{2}\sigma(e_i)\rangle}{\langle e_3, :{+}kS_3\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_3))(a)\rangle \qquad \langle e_0e_1e_2, :\text{-}k, \bigcup_{i=0}^{2}\sigma(e_i)\rangle}\text{merge-3}$$

$$\frac{\langle e_3, :{+}kS_3\text{-}k, \lambda a\,.\,\bigcup(\sigma(e_3))(a)\rangle \qquad \langle e_0e_1e_2, :\text{-}k, \bigcup_{i=0}^{2}\sigma(e_i)\rangle}{\langle e_0e_1e_2e_3, :S_3\text{-}k, \bigcup(\sigma(e_3))(\bigcup_{i=0}^{2}\sigma(e_i))\rangle}\text{move-1.}$$

$$\textit{End}: \quad \frac{\langle e_4, ::=S_3{+}kS_4, \lambda a\,.\,\bigcup(\sigma(e_4))(a))\rangle \qquad \langle e_0e_1e_2e_3, :S_3\text{-}k, \bigcup_{i=0}^{3}\sigma(e_i)\rangle}{\langle e_4, :{+}kS_4, \lambda a\,.\,\bigcup(\sigma(e_4))(a))\rangle \qquad \langle e_0e_1e_2e_3, :\text{-}k, \bigcup_{i=0}^{3}\sigma(e_i)\rangle}\text{merge-3}$$

$$\frac{\langle e_4, :{+}kS_4, \lambda a\,.\,\bigcup(\sigma(e_4))(a)\rangle \qquad \langle e_0e_1e_2e_3, :\text{-}k, \bigcup_{i=0}^{3}\sigma(e_i)\rangle}{\langle e_0e_1e_2e_3e_4, :S_4, \bigcup(\sigma(e_4))(\bigcup_{i=0}^{3}\sigma(e_i))\rangle}\text{move-1.}$$

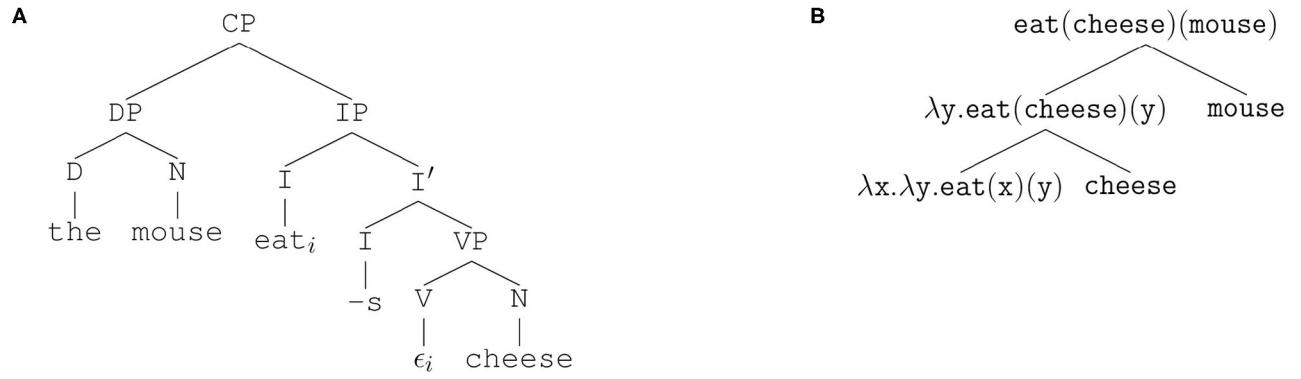

FIGURE 2 | Generative grammar analysis of example UMP (Equation 2). **(A)** Syntactic phrase structure tree. **(B)** Semantic tree from lambda calculus.

argument position for the new set element to be integrated is opened and closed again by consecutive λ-applications after the current feature-value pair has been added. Finally, the formalism replies with a tuple $\tau_{obs} \in Tup(Situation)$.

The last compositional step generates the linguistic sign $\langle \mathtt{e}_0\mathtt{e}_1\mathtt{e}_2\mathtt{e}_3\mathtt{e}_4,\ \ :\mathtt{S}_4, \bigcup_{i=0}^{4} \sigma(e_i)\rangle$. The associated semantics corresponds to the structure of tuples, as defined for relational schemes according to Equation (6). If this formalism is applied to the relational scheme $R(Situation)$ (for relation scheme definitions see section 5.4) it results to the semantic representation of the observed situation

$$\tau_{obs} = \bigcup_{i=0}^{n+1} \sigma(e_i) = \{X \to x \in \mathrm{dom}(X), Y \to y \in \mathrm{dom}(Y), \mathcal{O} \to o \in \mathrm{dom}(\mathcal{O})\}. \quad (17)$$

The observation tuple τ_{obs} contains the set of disordered feature-value-mappings, which corresponds to the translation result of the interpretation. Based on this tuple the next action is selected and a further perception-action-cycle is initiated.

4.2. Articulation

After the action decision by the behavior control (see **Figure 1**), the MG formalism must be applied to a semantic representation for the action in question. To this end, the formalism is fed with an action tuple $\tau_{act} \in Tup(Action)$. Each action tuple is initially given in the form.

$$\tau_{act} = \{\Delta X \to \Delta x \in \mathrm{dom}(\Delta X), \Delta Y \to \Delta y \in \mathrm{dom}(\Delta Y)\}. \quad (18)$$

In order to generate an actuator instruction $\mathbf{a} = (\mathtt{e}_0, \mathtt{e}_1, \mathtt{e}_2, \mathtt{e}_3)$ or $\mathbf{a} = (\texttt{<start>}, \Delta\mathtt{x}, \Delta\mathtt{y}, \texttt{<end>})$, respectively, the feature value pairs of the relational scheme $R(Action)$ must be mapped to the associated linguistic signs. These signs are stored in the linguistic lexicon of the articulation (analogous to the interpretation). In this case too, the formalism alternately switches between the rules merge-3 and move-1. The difference to the interpretation is that the formalism is now fed with a *set of linguistic signs* and responds with a sequence of symbols. In our mouse-maze scenario this set comprises the following elements: $\langle \mathtt{e}_0, \mathtt{::S_0\text{-}k}, \emptyset\rangle$, $\langle \mathtt{e}_3, \mathtt{::=S_2{+}kS_3}, \emptyset\rangle$ as well as $\langle \mathtt{e}_1, \mathtt{::=S_0{+}k\ \ S_1\text{-}k}, \sigma(e_1)\rangle$ and $\langle \mathtt{e}_2, \mathtt{::=S_1{+}kS_2\text{-}k}, \sigma(e_2)\rangle$. Applying the production rules of a regular grammar, the derivation would result to the following rule sequence: $\mathtt{S}_0 \to \mathtt{e}_0\mathtt{S}_1$, $\mathtt{S}_1 \to \mathtt{e}_1\mathtt{S}_2$, $\mathtt{S}_2 \to \mathtt{e}_2\mathtt{S}_3$ and $\mathtt{S}_3 \to \mathtt{e}_3$. This results in $\mathtt{S}_0 \to \mathtt{e}_0\mathtt{e}_1\mathtt{e}_2\mathtt{e}_3$, in which the root of the derivation tree $\mathtt{S}_0$ comprises the whole action sequence.

The MG-formalism preserves this order through the selectors of the associated exponents: After the formalism is starting with the sign $\langle \mathtt{e}_0, \mathtt{::S_0\text{-}k}, \emptyset\rangle$ the exponent $\mathtt{e}_1$ is required by the selector "$=\mathtt{S}_0$" of the sign $\langle \mathtt{e}_1, \mathtt{::=S_0{+}kS_1\text{-}k}, \sigma(e_1)\rangle$. Next, the exponent $\mathtt{e}_2$ is required by the selector "$=\mathtt{S}_1$" of the sign $\langle \mathtt{e}_2, \mathtt{::=S_1{+}kS_2\text{-}k}, \sigma(e_2)\rangle$. Finally, the exponent $\mathtt{e}_3$ is required by the selector "$=\mathtt{S}_2$" of the remaining sign in the articulation set $\langle \mathtt{e}_3, \mathtt{::=S_2{+}kS_3}, \emptyset\rangle$. The complete concatenated string $\mathbf{a} = (\mathtt{e}_0, \mathtt{e}_1, \mathtt{e}_2, \mathtt{e}_3) = (\texttt{<start>}, \Delta\mathtt{x}, \Delta\mathtt{y}, \texttt{<end>})$ is inferred then by the last move-1 rule and corresponds to the action sequence that is subsequently transmitted to the agent's actuators.

5. LINGUISTIC COMMUNICATION

As in the case of physical interaction, we also describe linguistic communication through a minimalist grammar. However, while interaction suffices with a minimalist implementation of regular grammars, exploiting only merge-3 and move-1 operations for the processing of linear time series of observation and actuation, natural language processing requires the full complexity of the MG formalism.

5.1. Utterance-Meaning Transducer (UMT)

In this section we propose a bidirectional *Utterance-Meaning-Transducer* (UMT) for both speech understanding and speech production by means of MG. For further illustrating the rules (10–11) and their applicability, let us stick with the example UMP (2) given in Sect. 2. Its syntactic analysis in terms of generative grammar (Haegeman, 1994) yields the (simplified) phrase structure tree in **Figure 2A**[4].

[4]For the sake of simplicity we refrain from presenting full-fledged X-bar hierarchies (Haegeman, 1994).

The syntactic categories in **Figure 2A** 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 **Figure 2B** using the notation from Sect. 3.5 together with the lambda calculus of the binary predicate in its Schönfinkel-Curry notation (Schönfinkel, 1924; Lohnstein, 2011).

Guided by the linguistic analyses in **Figure 2**, an expert could construe a minimalist lexicon as given in **Table 2** by hand (Stabler and Keenan, 2003).

$$\frac{\langle \texttt{the}, \texttt{::=n d -k}, \epsilon\rangle \quad \langle \texttt{mouse}, \texttt{::n}, \texttt{mouse}\rangle}{\langle \texttt{the mouse}, \texttt{:d -k}, \texttt{mouse}\rangle}\,\text{merge-1} \tag{19-1}$$

$$\frac{\langle \texttt{eat}, \texttt{::=n v -f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle \quad \langle \texttt{cheese}, \texttt{::n -k}, \texttt{cheese}\rangle}{\langle \texttt{eat}, \texttt{:v -f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{cheese}, \texttt{:-k}, \texttt{cheese}\rangle}\,\text{merge-3} \tag{19-2}$$

$$\frac{\langle \epsilon, \texttt{::=v +k =d pred}, \lambda P.\lambda Q.\texttt{Q(P)}\rangle \quad \langle \texttt{eat}, \texttt{:v -f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{cheese}, \texttt{:-k}, \texttt{cheese}\rangle}{\langle \epsilon, \texttt{:+k =d pred}, \lambda P.\lambda Q.\texttt{Q(P)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{cheese}, \texttt{:-k}, \texttt{cheese}\rangle}\,\text{merge-3} \tag{19-3}$$

$$\frac{\langle \epsilon, \texttt{:+k =d pred}, \lambda P.\lambda Q.\texttt{Q(P)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{cheese}, \texttt{:-k}, \texttt{cheese}\rangle}{\langle \texttt{cheese}, \texttt{:=d pred}, (\lambda P.\lambda Q.\texttt{Q(P)})(\texttt{cheese})\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle}\,\text{move-1} \tag{19-4}$$

$$\frac{\langle \texttt{cheese}, \texttt{:=d pred}, (\lambda P.\lambda Q.\texttt{Q(P)})(\texttt{cheese})\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle}{\langle \texttt{cheese}, \texttt{:=d pred}, \lambda Q.\texttt{Q(cheese)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle}\,\lambda\text{-app.} \tag{19-5}$$

$$\frac{\langle \texttt{cheese}, \texttt{:=d pred}, \lambda Q.\texttt{Q(cheese)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle \quad \langle \texttt{the mouse}, \texttt{:d -k}, \texttt{mouse}\rangle}{\langle \texttt{cheese}, \texttt{:pred}, \lambda Q.\texttt{Q(cheese)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}\,\text{merge-3} \tag{19-6}$$

$$\frac{\langle \texttt{-s}, \texttt{::=pred +f +k t}, \epsilon\rangle \quad \langle \texttt{cheese}, \texttt{:pred}, \lambda Q.\texttt{Q(cheese)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}{\langle \texttt{-s cheese}, \texttt{:+f +k t}, \lambda Q.\texttt{Q(cheese)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}\,\text{merge-1} \tag{19-7}$$

$$\frac{\langle \texttt{-s cheese}, \texttt{:+f +k t}, \lambda Q.\texttt{Q(cheese)}\rangle\langle \texttt{eat}, \texttt{:-f}, \lambda x.\lambda y.\texttt{eat(x)(y)}\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}{\langle \texttt{eat-s cheese}, \texttt{:+k t}, (\lambda Q.\texttt{Q(cheese)})(\lambda x.\lambda y.\texttt{eat(x)(y)})\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}\,\text{move-1} \tag{19-8}$$

$$\frac{\langle \texttt{eats cheese}, \texttt{:+k t}, (\lambda Q.\texttt{Q(cheese)})(\lambda x.\lambda y.\texttt{eat(x)(y)})\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}{\langle \texttt{eats cheese}, \texttt{:+k t}, (\lambda x.\lambda y.\texttt{eat(x)(y)})(\texttt{cheese})\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}\,\lambda\text{-app.} \tag{19-9}$$

$$\frac{\langle \texttt{eats cheese}, \texttt{:+k t}, (\lambda x.\lambda y.\texttt{eat(x)(y)})(\texttt{cheese})\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}{\langle \texttt{eats cheese}, \texttt{:+k t}, \lambda y.\texttt{eat(cheese)(y)}\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}\,\lambda\text{-app.} \tag{19-10}$$

$$\frac{\langle \texttt{eats cheese}, \texttt{:+k t}, \lambda y.\texttt{eat(cheese)(y)}\rangle\langle \texttt{the mouse}, \texttt{:-k}, \texttt{mouse}\rangle}{\langle \texttt{the mouse eats cheese}, \texttt{:t}, (\lambda y.\texttt{eat(cheese)(y)})(\texttt{mouse})\rangle}\,\text{move-1} \tag{19-11}$$

$$\frac{\langle \texttt{the mouse eats cheese}, \texttt{: t}, (\lambda y.\texttt{eat(cheese)(y)})(\texttt{mouse})\rangle}{\langle \texttt{the mouse eats cheese}, \texttt{:t}, \texttt{eat(cheese)(mouse)}\rangle}\,\lambda\text{-app.} \tag{19-12}$$

$$\frac{\langle \epsilon, \texttt{::=t c}, \epsilon\rangle \quad \langle \texttt{the mouse eats cheese}, \texttt{:t}, \texttt{eat(cheese)(mouse)}\rangle}{\langle \texttt{the mouse eats cheese}, \texttt{:c}, \texttt{eat(cheese)(mouse)}\rangle}\,\text{merge-1}\,. \tag{19-13}$$

We adopt a shallow semantic model, where the universe of discourse only contains two individuals, the mouse and a piece of cheese[5]. Then, the lexicon (**Table 2**) 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 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 bound by two lambda expressions λx.λy. 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 (**Table 2**), the sign (16) is obtained by the minimalist derivation (19).

In the first step, (19-1), 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 (19-4) 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, (19-7), the present-tense suffix unifies with the predicate, entailing an inflection

[5]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.

TABLE 2 | UMT minimalist lexicon for example grammar (**Figure 2**).

⟨mouse, ::n, mouse⟩	⟨cheese, ::n -k, cheese⟩
⟨the, ::=n d -k, ϵ⟩	⟨eat, ::=n v -f, λx.λy.eat(x)(y)⟩
⟨-s, ::=pred +f +k t, ϵ⟩	⟨ϵ, ::=v +k =d pred, λP.λQ.Q(P)⟩
⟨ϵ, ::=t c, ϵ⟩	

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 **Figure 2B**. Step 11 assigns nominative case to the subject through movement into specifier 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.

Derivations such as (19) are essential for MG. 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 (Harkema, 2001; Mainguy, 2010; Stabler, 2011b; Stanojević and Stabler, 2018). 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 (Hale, 2011). 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 (2011b)'s top-down recognizer as outlined earlier by beim Graben et al. (2019a). Its generalization toward the recent left-corner parser (Stanojević and Stabler, 2018) is straightforward.

The central object for MG language processing is the *derivation tree* obtained from a bottom-up derivation as in (19). **Figure 3** 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 (2011b)'s algorithm.

Pursuing the tree paths in **Figure 3** from the bottom to the top, provides exactly the derivation (19). However, reading it from the top toward the bottom allows for an interpretation in terms of *multiple context-free grammars* (Seki et al., 1991; Michaelis, 2001) (MCFG) where categories are n-ary predicates over string exponents. Like in context-free grammars, every branching in the derivation tree (**Figure 3**) leads to one phrase structure rule in the MCFG. Thus, the MCFG enumerated in (20) codifies the MG (**Table 2**).

$$\langle \texttt{:c}\rangle(e_0e_1) \leftarrow \langle \texttt{::=t c}\rangle(e_0) \quad \langle \texttt{:t}\rangle(e_1) \tag{20-1}$$

$$\langle \texttt{:t}\rangle(e_1e_0) \leftarrow \langle \texttt{:+k t, -k}\rangle(e_0,e_1) \tag{20-2}$$

$$\langle \texttt{:+k t, -k}\rangle(e_1e_0,e_2) \leftarrow \langle \texttt{:+f +k t, -f, -k}\rangle(e_0,e_1,e_2) \tag{20-3}$$

$$\langle \texttt{:+f +k t, -f, -k}\rangle(e_0e_1,e_2,e_3) \leftarrow \langle \texttt{::=pred +f +k t}\rangle(e_0)$$

$$\langle \texttt{:pred, -f, -k}\rangle(e_1,e_2,e_3) \tag{20-4}$$

$$\langle \texttt{:pred, -f, -k}\rangle(e_0,e_1,e_2) \leftarrow \langle \texttt{:=d pred, -f}\rangle(e_0,e_1) \quad \langle \texttt{:d -k}\rangle(e_2) \tag{20-5}$$

$$\langle \texttt{:=d pred, -f}\rangle(e_2e_0,e_1) \leftarrow \langle \texttt{:+k =d pred, -f, -k}\rangle(e_0,e_1,e_2) \tag{20-6}$$

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

$$\langle \texttt{:v -f, -k}\rangle(e_1,e_2) \tag{20-7}$$

$$\langle \texttt{:v -f, -k}\rangle(e_0,e_1) \leftarrow \langle \texttt{::=n v -f}\rangle(e_0) \quad \langle \texttt{::n -k}\rangle(e_1) \tag{20-8}$$

$$\langle \texttt{:d -k}\rangle(e_0e_1) \leftarrow \langle \texttt{::=n d -k}\rangle(e_0) \quad \langle \texttt{::n}\rangle(e_1) \tag{20-9}$$

$$\langle \texttt{::n}\rangle(\texttt{mouse}) \tag{20-10}$$

$$\langle \texttt{::n -k}\rangle(\texttt{cheese}) \tag{20-11}$$

$$\langle \texttt{::=n d -k}\rangle(\texttt{the}) \tag{20-12}$$

$$\langle \texttt{::=n v -f}\rangle(\texttt{eat}) \tag{20-13}$$

$$\langle \texttt{::=pred +f +k t}\rangle(\texttt{-s}) \tag{20-14}$$

$$\langle \texttt{::=v +k =d pred}\rangle(\epsilon) \tag{20-15}$$

$$\langle \texttt{::=t c}\rangle(\epsilon) \tag{20-16}$$

In (20), 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 (20-1–20-9) are directly obtained from the derivation tree (**Figure 3**) by reverting the merge and move operations of (19) through their "unmerge" and "unmove" counterparts (Harkema, 2001). The MCFG axioms, i. e. the lexical rules (20-10 – 20-16), are reformulations of the entries in the MG lexicon (**Table 2**).

The UMT's language production module finds a semantic representation of an intended utterance in form of a Schönfinkel-Curry (Schönfinkel, 1924; Lohnstein, 2011) formula of predicate logic, such as eat(cheese)(mouse), for instance. According to **Figure 2B** this is a hierarchical data structure that can control the MG derivation (19). Thus, the cognitive agent accesses its mental lexicon, either through **Table 2** or its MCFG twin (20) in order to retrieve the linguistic signs for the denotations eat, cheese, and mouse. Then, the semantic tree (**Figure 2B**) governs the correct derivation (19) 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 the mouse eats cheese is uttered.

The language understanding module of our UMT, by contrast, 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 (2011b)'s priority queue top-down parser. 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 3** presents the temporal

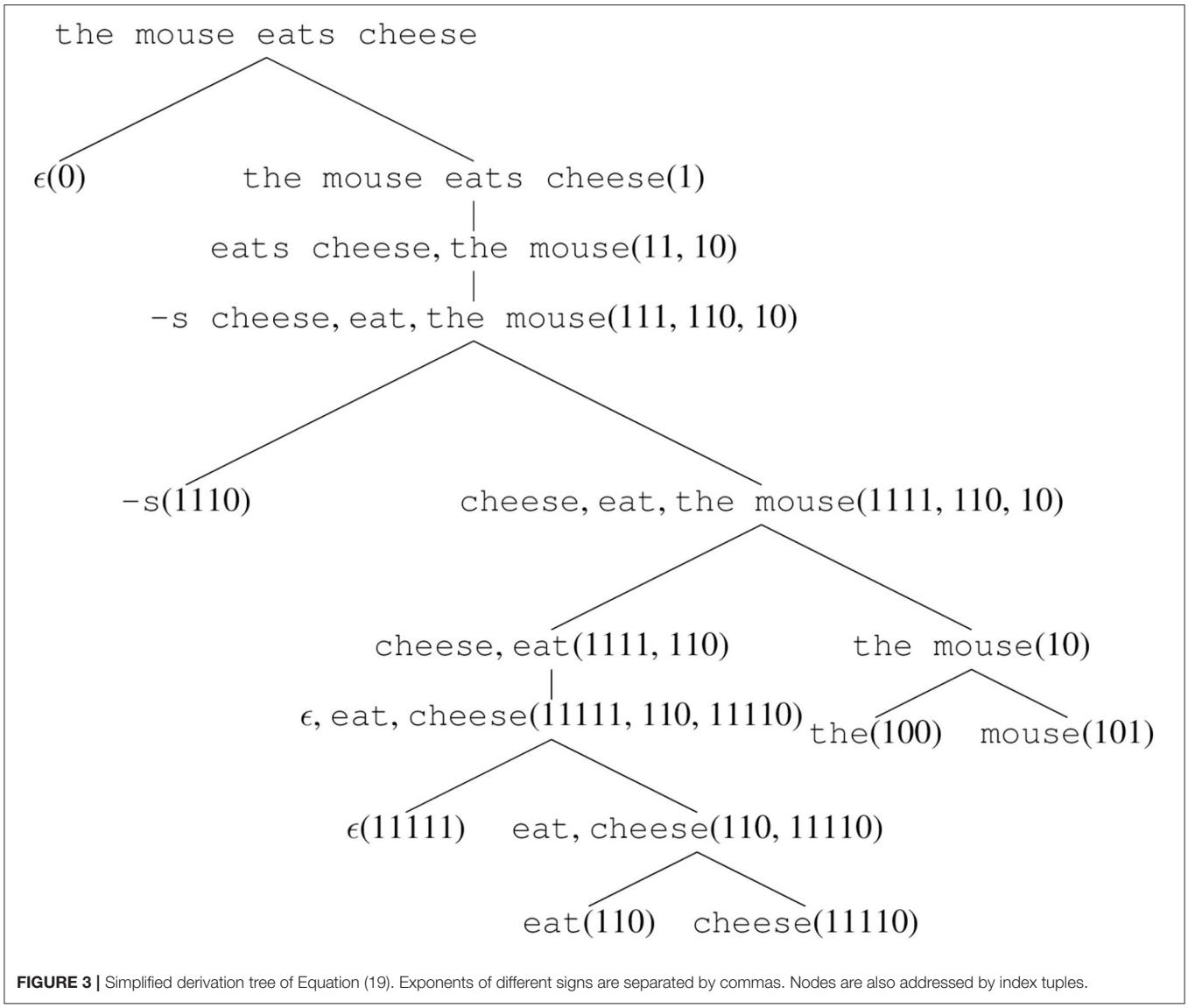

FIGURE 3 | Simplified derivation tree of Equation (19). Exponents of different signs are separated by commas. Nodes are also addressed by index tuples.

evolution of the top-down recognizer's configurations while processing the utterance `the mouse eats cheese`.

The parser is initialized with the input string to be processed and the MCFG start symbol ⟨`:c`⟩(ϵ)—corresponding to the MG start symbol `c`—at the top of the priority queue. For each rule of the MCFG (20), the algorithm replaces its string variables by an index tuple that addresses the corresponding nodes in the derivation tree (**Figure 3**) (Stabler, 2011b). 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 (20) become ordered according to their temporal appearance in the utterance. Using the notation of the derivation tree (**Figure 3**), 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 (**Figure 3**) reveals that the index length correlates with the tree depth. In our example, the items ⟨`::n -k`⟩(11110) and ⟨`::=v +k =d pred`⟩(11111) in the priority queue have

TABLE 3 | MG top-down parse of `the mouse eats cheese`.

Step	Input	Syntactic queue	Operation
1.	the mouse eats cheese	⟨:c⟩(ϵ)	expand (20-1)
2.	the mouse eats cheese	⟨::=t c⟩(0)⟨:t⟩(1)	scan (20-16)
3.	the mouse eats cheese	⟨:t⟩(1)	expand (20-2)
4.	the mouse eats cheese	⟨:+k t, -k⟩(11, 10)	expand (20-3)
5.	the mouse eats cheese	⟨:+f +k t, -f, -k⟩(111, 110, 10)	expand (20-4)
6.	the mouse eats cheese	⟨::=pred +f +k t⟩(1110)⟨:pred, -f, -k⟩(1111, 110, 10)	sort
7.	the mouse eats cheese	⟨:pred, -f, -k⟩(1111, 110, 10)⟨::=pred +f +k t⟩(1110)	expand (20-5)
8.	the mouse eats cheese	⟨:=d pred, -f⟩(1111, 110)⟨:d -k⟩(10)⟨::=pred +f +k t⟩(1110)	sort
9.	the mouse eats cheese	⟨:d -k⟩(10)⟨:=d pred, -f⟩(1111, 110)⟨::=pred +f +k t⟩(1110)	expand (20-9)
10.	the mouse eats cheese	⟨::=n d -k⟩(100)⟨::n⟩(101)⟨:=d pred, -f⟩(1111, 110)⟨::=pred +f +k t⟩(1110)	scan (20-12)
11.	mouse eats cheese	⟨::n⟩(101)⟨:=d pred, -f⟩(1111, 110)⟨::=pred +f +k t⟩(1110)	scan (20-10)
12.	eats cheese	⟨:=d pred, -f⟩(1111, 110)⟨::=pred +f +k t⟩(1110)	expand (20-6)
13.	eats cheese	⟨:+k =d pred, -f, -k⟩(11111, 110, 11110)⟨::=pred +f +k t⟩(1110)	expand (20-7)
14.	eats cheese	⟨::=v +k =d pred⟩(11111)⟨:v -f, -k⟩(110, 11110)⟨::=pred +f +k t⟩(1110)	sort
15.	eats cheese	⟨:v -f, -k⟩(110, 11110)⟨::=pred +f +k t⟩(1110)⟨::=v +k =d pred⟩(11111)	expand (20-8)
16.	eats cheese	⟨::=n v -f⟩(110)⟨::n -k⟩(11110)⟨::=pred +f +k t⟩(1110)⟨::=v +k =d pred⟩(11111)	sort
17.	eats cheese	⟨::=n v -f⟩(110)⟨::=pred +f +k t⟩(1110)⟨::n -k⟩(11110)⟨::=v +k =d pred⟩(11111)	scan (20-13)
18.	-s cheese	⟨::=pred +f +k t⟩(1110)⟨::n -k⟩(11110)⟨::=v +k =d pred⟩(11111)	scan (20-14)
19.	cheese	⟨::n -k⟩(11110)⟨::=v +k =d pred⟩(11111)	scan (20-11)
20.	ϵ	⟨::=v +k =d pred⟩(11111)	scan (20-15)
21.	ϵ	ϵ	accept

TABLE 4 | Semantic processing of `the mouse eats cheese`.

Step	Input	Semantic queue	Operation
1.	the mouse eats cheese	ϵ	scan (20-16)
2.	the mouse eats cheese	⟨ϵ⟩(0)	apply
3.	the mouse eats cheese	ϵ	scan (20-12)
4.	mouse eats cheese	⟨ϵ⟩(100)	apply
5.	mouse eats cheese	ϵ	scan (20-10)
6.	eats cheese	⟨mouse⟩(101)	scan (20-13)
7.	-s cheese	⟨mouse⟩(101)⟨λx.λy.eat(x)(y)⟩(110)	scan (20-14)
8.	cheese	⟨mouse⟩(101)⟨λx.λy.eat(x)(y)⟩(110)⟨ϵ⟩(1110)	apply
9.	cheese	⟨mouse⟩(101)⟨λx.λy.eat(x)(y)⟩(110)	scan (20-11)
10.	ϵ	⟨mouse⟩(101)⟨λx.λy.eat(x)(y)⟩(110)⟨cheese⟩(11110)	scan (20-15)
11.	ϵ	⟨mouse⟩(101)⟨λx.λy.eat(x)(y)⟩(110)⟨cheese⟩(11110)⟨λP.λQ.Q(P)⟩(11111)	sort
11.	ϵ	⟨λP.λQ.Q(P)⟩(11111)⟨cheese⟩(11110)⟨λx.λy.eat(x)(y)⟩(110)⟨mouse⟩(101)	apply
12.	ϵ	⟨λQ.Q(cheese)⟩(1111)⟨λx.λy.eat(x)(y)⟩(110)⟨mouse⟩(101)	apply
13.	ϵ	⟨(λx.λy.eat(x)(y))(cheese)⟩(111)⟨mouse⟩(101)	apply
14.	ϵ	⟨λy.eat(cheese)(y)⟩(111)⟨mouse⟩(101)	apply
15.	ϵ	⟨eat(cheese)(mouse)⟩(11)	understand

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 (19-5). Moreover, also the semantic analysis in **Figure 2B** 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 4** illustrates the semantic parsing for the given example.

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

TABLE 5 | Minimalist lexicon–communication.

⟨contains, ::=n =n v, λq.λp.contain(q)(p)⟩	⟨cheese, ::n, cheese⟩
⟨lies, ::=p =n v, λq.λp.lie(q)(p)⟩	⟨field, ::n, field⟩
⟨(x,y), ::p -f, ⟨x, y⟩⟩	⟨ϵ, ::=n =p m -g, λP.λp.P(p)⟩
⟨ϵ, ::=m +f +g n, λP.λQ.QP⟩	⟨in, ::=n p, ϵ⟩

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 upon 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 *understand*ing state.

5.2. Interpretation

After preparing the UMT, we come back to our particular mouse-maze example. As previously explained, the agent could find its target objects by physically exploring its state space through interaction. Yet, another possibility is linguistic communication where an operator may inform the agent about the correct position of a target. For this case, we present another MG in **Table 5**. It contains the linguistic signs that can be used to compose the meaning of a message.

Our "communication" lexicon comprises two verbs, `contains` and `lies`, of the basic type `v`, which we want to consider here as the start symbol of the MG. The verb `contains` is transitive and therefore corresponds to a binary predicate, as indicated above by the lambda term in the Schönfinkel representation (Lohnstein, 2011). Syntactically, this binary predicate is expressed by two selectors `=n`, whereas in linguistics we have to take into account, that a transitive verb first selects a direct object and subsequently its subject (both nouns of the type `n` "noun"). The verb `lies` is intransitive, but can be modified by a prepositional phrase through a selector (`=p`). There are two nouns in the lexicon: `field` and `cheese`, their semantics are the corresponding individual names. The field coordinates (x, y) in the labyrinth are expressed by the exponent `(x,y)`, whose syntactic function is an adjunct of the basic type `p` (prepositional phrase). This must be moved once in the course of the structure development, for which the licensee `-f` is intended. The adjunction itself is triggered by a phonetically empty entry of the basic type `m` (modifier). This has two selectors: `=n` selects the noun phrase to be modified, while `=p` selects the adjoint prepositional phrase. Then `-g` licenses a movement. The semantics of the adjunction here expressed in a simplified manner as the predication `λP.λp.P(p)`. That is, there is an object `p` which has the property `P`. Another phonetically empty expression selects a modified noun phrase (`=m`), then licenses two movements (`+f +g`) and generates itself a noun phrase of the base type `n`. Semantically, an argument movement is canceled again by the exchange operator `λP.λQ.QP`. Finally, there is a preposition `in` of the basic type `p`, which also selects a noun phrase for adjunction (`=n`). Instead of formalizing their locative semantics in concrete terms, we simplify them with an empty lambda expression ϵ. In order to reduce the effort for the exploration phase, we can send the following message "`field (x,y) contains cheese`" to the mouse. This utterance can be processed by our UMT above, for which we present the minimalist bottom-up derivation as follows.

$$\frac{\langle \texttt{contains}, \texttt{::=n =n v}, \lambda\texttt{q}.\lambda\texttt{p}.\texttt{contain(q)(p)}\rangle \quad \langle \texttt{cheese}, \texttt{::n}, \texttt{cheese}\rangle}{\langle \texttt{contains cheese}, \texttt{:=n v}, (\lambda\texttt{q}.\lambda\texttt{p}.\texttt{contain(q)(p)})(\texttt{cheese})\rangle}\text{merge-1}$$

$$\frac{\langle \epsilon, \texttt{::=n =p m -g}, \lambda\texttt{P}.\lambda\texttt{p}.\texttt{P(p)}\rangle \quad \langle \texttt{field}, \texttt{::n}, \texttt{field}\rangle}{\langle \texttt{field}, \texttt{:=p m -g}, (\lambda\texttt{P}.\lambda\texttt{p}.\texttt{P(p)})(\texttt{field})\rangle}\text{merge-1}$$

$$\frac{\langle \texttt{field}, \texttt{::=p m -g}, \lambda\texttt{p}.\texttt{field(p)}\rangle \quad \langle \texttt{(x,y)}, \texttt{:p -f}, \langle \texttt{x}, \texttt{y}\rangle\rangle}{\langle \texttt{field}, \texttt{:m -g}, \lambda\texttt{p}.\texttt{field(p)}\rangle\langle \texttt{(x,y)}, \texttt{:-f}, \langle \texttt{x}, \texttt{y}\rangle\rangle}\text{merge-3}$$

$$\frac{\langle \epsilon, \texttt{::=m +f +g n}, \lambda\texttt{P}.\lambda\texttt{Q}.\texttt{QP}\rangle \quad \langle \texttt{field}, \texttt{:m -g}, \lambda\texttt{p}.\texttt{field(p)}\rangle\langle \texttt{(x,y)}, \texttt{:-f}, \langle \texttt{x}, \texttt{y}\rangle\rangle}{\langle \epsilon, \texttt{:+f +g n}, \lambda\texttt{P}.\lambda\texttt{Q}.\texttt{QP}\rangle\langle \texttt{field}, \texttt{:-g}, \lambda\texttt{p}.\texttt{field(p)}\rangle\langle \texttt{(x,y)}, \texttt{:-f}, \langle \texttt{x}, \texttt{y}\rangle\rangle}\text{merge-3}$$

$$\frac{\langle \epsilon, \texttt{:+f +g n}, \lambda\texttt{P}.\lambda\texttt{Q}.\texttt{QP}\rangle\langle \texttt{field}, \texttt{:-g}, \lambda\texttt{p}.\texttt{field(p)}\rangle\langle \texttt{(x,y)}, \texttt{:-f}, \langle \texttt{x}, \texttt{y}\rangle\rangle}{\langle \texttt{(x,y)}, \texttt{:+g n}, (\lambda\texttt{P}.\lambda\texttt{Q}.\texttt{QP})(\langle \texttt{x}, \texttt{y}\rangle)\rangle\langle \texttt{field}, \texttt{:-g}, \lambda\texttt{p}.\texttt{field(p)}\rangle}\text{move-1}$$

$$\frac{\langle \texttt{(x,y)}, \texttt{:+g n}, \lambda\texttt{Q}.\texttt{Q}(\langle \texttt{x}, \texttt{y}\rangle)\rangle\langle \texttt{field}, \texttt{:-g}, \lambda\texttt{p}.\texttt{field(p)}\rangle}{\langle \texttt{field (x,y)}, \texttt{:n}, (\lambda\texttt{Q}.\texttt{Q}(\langle \texttt{x}, \texttt{y}\rangle))(\lambda\texttt{p}.\texttt{Feld(p)})\rangle}\text{move-1}$$

$$\frac{\langle \texttt{contains cheese}, \texttt{:=n v}, \lambda\texttt{p}.\texttt{contain(cheese)(p)}\rangle \quad \langle \texttt{field (x,y)}, \texttt{:n}, \texttt{field}(\langle \texttt{x}, \texttt{y}\rangle)\rangle}{\langle \texttt{field (x,y) contains cheese}, \texttt{:v}, (\lambda\texttt{p}.\texttt{contain(cheese)(p)})(\texttt{field}(\langle \texttt{x}, \texttt{y}\rangle))\rangle}\text{merge-2}$$

Thereby, the utterance "`field (x,y) contains cheese`" is recognized in the exponent, so that its semantics can be interpreted in terms of model theory as a predicate logic formula after a last lambda application (λ`p.contain(cheese)(p))(field((x,y))) = contain(cheese)(field((x,y)))`. The denotation of this predicate assignment results to

$$[\texttt{contain(cheese)(field((x,y)))}] = ([\texttt{cheese}], [\texttt{field((x,y))}]) \in [\texttt{contain}], \quad (21)$$

This corresponds to a representational data structure which can be inserted in an associated relational scheme.

5.3. Articulation

For speech production, we start from the fact "`field (x,y) contains cheese`" and show the possibility of stylistic creativity. First, we note that the denotation of `contain` is coextensive with that of `lie`, i. e. `[contain] = [lie]` applies. Based on this capability an agent is able to express what it has understood in its own words. In this way, the adjustment of common ideas or concepts required for successful communication can take place. If the agent wants to articulate such an utterance, it must first translate the facts into a predicate logic formula and has to consider, that the order of arguments of the subject and the direct object must be swapped: `lie(field((x,y)))(cheese)`. A query in the minimalist lexicon then leads to the following derivation, whereby we reuse the previously derived expression $\langle$`field (x,y), :n, field((x,y))`$\rangle$ in the sense of dynamic programming.

$$\frac{\langle \texttt{in, ::=n p},\epsilon\rangle \quad \langle \texttt{field (x,y), :n, field((x,y))}\rangle}{\langle \texttt{in field (x,y), :p, field((x,y))}\rangle}\ \text{merge-1}$$

$$\frac{\langle \texttt{lies, ::=p =n v, }\lambda\texttt{q.}\lambda\texttt{p.lie(q)(p)}\rangle \quad \langle \texttt{in field (x,y), :p, field((x,y))}\rangle}{\langle \texttt{lies in field (x,y), :=n v, (}\lambda\texttt{q.}\lambda\texttt{p.lie(q)(p))(field((x,y)))}\rangle}\ \text{merge-1}$$

$$\frac{\langle \texttt{lies in field (x,y), :=n v, }\lambda\texttt{p.lie(field((x,y)))(p)}\rangle \quad \langle \texttt{cheese, ::n, cheese}\rangle}{\langle \texttt{cheese lies in field (x,y), :v, (}\lambda\texttt{p.lie(field((x,y)))(p))(cheese)}\rangle}\ \text{merge-2}$$

The present derivation finally comprises after a final lambda application the intended semantics (λ`p.lie(field((x,y)))(p))(cheese) = lie(field((x,y)))(cheese)` and leads also to the synonymous utterance "`cheese lies in field (x,y)`."

5.4. Knowledge Integration

In the previous sections we described how the information of symbolic sequences can be transformed into logically processable knowledge and vice versa by interaction or communication, respectively. In order to process knowledge, it has to be saved and retrieved again. These operations are essential characteristics of a universal algorithmic system with which calculation rules can be implemented. Based on such a knowledge processing system a cognitive agent is able to pursue goals, understand situations, select cost optimal actions and to predict its consequences. Further, the agent should be able to learn behavioral relations and to adapt to changing environmental conditions. These requirements can be satisfied more efficiently if the required knowledge is structured and available as a (dynamic) model of the external world. To this end, we apply the state space model, with which we can describe the relations $R_{Situation} \subseteq States \times \mathcal{O}$ and $R_{Movement} \subseteq States \times \mathcal{A} \times States$, where the set *States* corresponds to the set $\mathcal{Z}$ of the state space model. Based on these relations predictions about the next state and the associated observations can be made. Prediction errors can either be used to recognize changes in the environmental conditions or in the realization of the selected actions, which can initiate an adaptation or diagnosis process. The states $z \in \mathcal{Z}$ correspond to the maze fields and have a finer inner structure based on ordered pairs $z = (x, y) \in X \times Y$, which are associated with the two-dimensional coordinates of the maze. They can be measured by sensors. The same applies to the elements $a \in \mathcal{A}$, where the finer structure is given by $a = (\Delta x, \Delta y) \in \Delta X \times \Delta Y$. The elements $o \in \mathcal{O}$ correspond to the target object types, which can be identified by an object classifier.

In order to build a dynamic model, the agent must have the ability to analyze experimentally. In the course of this experimentation process (exploration), knowledge about "situations" and "movements" is stored in relational schemes as long as no contradictions occur. Such an exploration process results—mathematically speaking—in a relation $R_{\mathcal{X}} \in Rel(\mathcal{X})$, whereby the identified relation $R_{\mathcal{X}}$ can be expressed by its characteristic function (e.g., $f_{R_{Situation}} : X \times Y \times \mathcal{O} \rightarrow \{0, 1\}$), which assigns exactly one truth value to each tuple $\tau \in Tup(\mathcal{X})$. After completion of the exploration phase, the evaluation of the characteristic functions can be used to answer questions like "What is the case?" or "What does this observation mean?" in a truth-functional sense and therefore leads to a veridical model. Such a model enables the agent to gain knowledge through simulation or inference, which in turn can then be transfered to reality. To this end, the agent has to compare the representational data structure of the perception (e. g. $\tau_{obs} = \{X \rightarrow x \in \text{dom}(X), Y \rightarrow y \in \text{dom}(Y), \rightarrow o \in \text{dom}(\mathcal{O})\}$) with some of the learned representational data structures of the associated relational scheme:

$$f_{R_{\mathcal{X}}}(\tau) = \begin{cases} True, & \text{if} \tau \in R_{\mathcal{X}}, \\ False, & \text{if } \tau \notin R_{\mathcal{X}}. \end{cases}$$

If matches are found, the agent understands the incoming information. In semiotics, this process is called denotation and can be realized by a logical compound system (see **Figure 4**). If no matches are found, an adaptation or coping process is triggered (Wolff et al., 2015).

In order to apply the dynamic model to the mouse-maze-application properly, we have to specify all attributes and relation schemes of states, situations, actions and movements as well as the corresponding relations that are required: *States* =

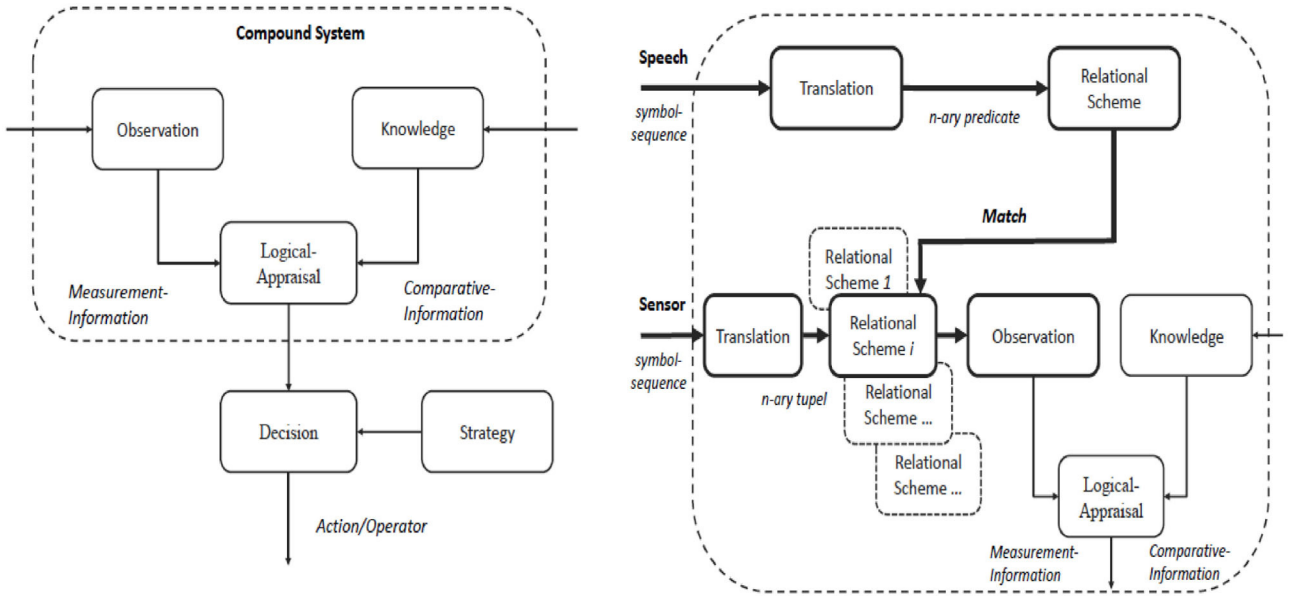

FIGURE 4 | Left: A logical compound system compares measurement information and model knowledge, subsequently a strategy based action is selected. Right: Merging of sensory information (interaction) and language information (communication) based on a double cognitive loop.

$\{X, Y\} \subset \mathcal{U}$, $R(States) = (X : \mathrm{dom}(X), Y : \mathrm{dom}(Y))$, $R_{States} \subseteq Tup(States)$, $Situation = \{States, \mathcal{O}\} \subset \mathcal{U}$, $R(Situation) = (X : \mathrm{dom}(X), Y : \mathrm{dom}(Y), \mathcal{O} : \mathrm{dom}(\mathcal{O}))$ and $R_{Situation} \subseteq Tup(Situation)$. $Action = \{\Delta X, \Delta Y\} \subset \mathcal{U}$, $R(Action) = (\Delta X : \mathrm{dom}(\Delta X), \Delta Y : \mathrm{dom}(\Delta Y))$ and $R_{Action} \subseteq Tup(Action)$. $Movement = \{States, Action\} \subset \mathcal{U}$, $R(Movement) = (States : \mathrm{dom}(States), Action : \mathrm{dom}(Action), States' : dom(States))$ and $R_{Movement} \subseteq Tup(Movement)$.

With these definitions suitable representational data structures are provided that are needed for the transformation and integration processes. However, despite the uniform description of the two cognitive loops using linguistic means, there is an essential difference between interaction and communication. Verbal communication operates primarily with object types and refers to predicate symbols as well. In contrast, the perceptive part of interaction is based on sensors and classifiers only, so that no relations between any object types can be directly encoded (Hausser, 2014). Hence, such relations have to be learned by the agent (Wolff et al., 2018). Yet, in a shared environment these different kinds of information can be used to solve the context problem. For this end, we consider the following example, in which the linguistic message "`field(x,y) contains cheese`" is translated into the binary predicate `contain(field(x,y),cheese)`. In order to obtain the truth value of this statement, it must be checked whether an instance of the object type "`cheese`" is located at the specified position $z = (x, y)$. That means the agent has to move to this position and take measurements. Then, the corresponding symbol sequence must be transformed into the tuple $\tau_{obs} = \{X \rightarrow x \in dom(X), Y \rightarrow y \in dom(Y), \mathcal{O} \rightarrow o \in dom(\mathcal{O})\} \in R_{Situation}$, which can now be compared with the entries of the relation $R_{contain} \subseteq \times X \times Y \times \mathcal{O}$. Please note that this relation is associated to the translated predicate and that an interpretation of the sensor values appropriate to the context is only possible if the proper scheme has been selected (**Figure 4**). Since both relations are designated by different names, it must be ensured that their denotation is identical, i. e. $[\texttt{contain}] = [Situation]$ must apply.

6. LANGUAGE ACQUISITION

So far 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 (**Table 2**), 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 (Gold, 1967). The studies of Bonato and Retoré (2001), Kobele et al. (2002), and Stabler et al. (2003) demonstrated how MG can be acquired by positive examples from linguistic dependence graphs (Nivre, 2003; Boston et al., 2010). 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 (Klein and Manning, 2004).

In our terminology, such statistical learning methods only reveal 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

TABLE 6 | Learned minimalist lexicon X_1 at time $t = 1$.

⟨the mouse eats cheese, :c, eat(cheese)(mouse)⟩

through reinforcement learning (Skinner, 2015; Sutton and Barto, 2018).

The language learner is a cognitive agent $\mathfrak{L}$ in a state X_t, to be identified with $\mathfrak{L}$'s mental lexicon at training time t. At time $t = 0$, X_0 is initialized as a *tabula rasa* with the empty lexicon

$$X_0 \leftarrow \emptyset \tag{22}$$

and exposed to UMPs produced by a teacher $\mathfrak{T}$. Note that we assume $\mathfrak{T}$ presenting already complete UMPs and not singular utterances to $\mathfrak{L}$. Thus, we circumvent the *symbol grounding problem* of firstly assigning meanings σ to uttered exponents e (Harnad, 1990), which will be addressed in future research. Moreover, we assume that $\mathfrak{L}$ is instructed to reproduce $\mathfrak{T}$'s utterances based on its own semantic understanding. This provides a feedback loop and therefore applicability of reinforcement learning (Skinner, 2015; Sutton and Barto, 2018). For our introductory example, we adopt the simple semantic model from Sect. 5. In each iteration, the teacher utters an UMP that should be learned by the learner.

6.1. First Iteration

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

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

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

$$X_t \leftarrow X_{t-1} \cup \{\langle e_t, \texttt{:c}, \sigma_t \rangle\}, \tag{23}$$

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

In this way, the mental lexicon X_1 shown in **Table 6** has been acquired at time $t = 1$.

6.2. Second Iteration

Next, let the teacher be uttering another proposition

$$u_2 = \langle \texttt{the rat eats cheese}, \texttt{eat(cheese)(rat)} \rangle. \tag{24}$$

Looking at u_1, u_2 together, the agent's pattern matching module (cf. van Zaanen, 2001) is able to find similarities between exponents and semantics, underlined in Equation (25).

$$u_1 = \langle \underline{\texttt{the}}\ \texttt{mouse}\ \underline{\texttt{eats cheese}}, \underline{\texttt{eat(cheese)}}\texttt{(mouse)} \rangle \tag{25-1}$$

$$u_2 = \langle \underline{\texttt{the}}\ \texttt{rat}\ \underline{\texttt{eats cheese}}, \underline{\texttt{eat(cheese)}}\texttt{(rat)} \rangle. \tag{25-2}$$

TABLE 7 | Learned minimalist lexicon X_2 at time $t = 2$.

⟨the mouse, :d, mouse⟩	⟨the rat, :d, rat⟩
⟨eats cheese, :=d c, λy.eat(cheese)(y)⟩	

TABLE 8 | Revised minimalist lexicon X_{21}.

⟨the, ::=n d, ϵ⟩	⟨mouse, ::n, mouse⟩
⟨rat, ::n, rat⟩	⟨eats cheese, :=d c, λy.eat(cheese)(y)⟩

TABLE 9 | Learned minimalist lexicon X_3 at time $t = 3$.

⟨the, ::=n d, ϵ⟩	⟨mouse, ::n, mouse⟩
⟨rat, ::n, rat⟩	⟨cheese, ::n, cheese⟩
⟨carrot, ::n, carrot⟩	⟨eats, ::=n =d c, λx.λy.eat(x)(y)⟩

Thus, $\mathfrak{L}$ creates two distinct items for `the mouse` and `the rat`, respectively, and carries out lambda abstraction to obtain the updated lexicon X_2 in **Table 7**.

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 **Table 7**, 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 **Table 8** can be devised.

For closing the reinforcement cycle, $\mathfrak{L}$ is supposed to produce utterances upon its own understanding. Thus, we assume that $\mathfrak{L}$ wants to express the proposition `eat(cheese)(rat)`. According to our discussion in Sect. 5, 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 $\mathfrak{T}$.

6.3. Third Iteration

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

$$u_3 = \langle \texttt{the mouse eats carrot}, \texttt{eat(carrot)(mouse)} \rangle. \tag{26}$$

Now we have to compare u_3 with the lexicon entry for `eats cheese` in (27).

$$\langle \texttt{the mouse}\ \underline{\texttt{eats}}\ \texttt{carrot}, \underline{\texttt{eat}}\texttt{(carrot)(mouse)} \rangle \tag{27-1}$$

$$\langle \underline{\texttt{eats}}\ \texttt{cheese}, \texttt{:c}, \lambda\texttt{y}.\underline{\texttt{eat}}\texttt{(cheese)(y)} \rangle. \tag{27-2}$$

Another lambda abstraction entails the lexicon X_3 in **Table 9**.

Here, the learner assumes that `eats` is a simple, lexical category without having further evidence as in Sect. 5.

Since $\mathfrak{L}$ 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 (28) that is rewarded by $\mathfrak{T}$.

TABLE 10 | Learned minimalist lexicon X_4 at time $t = 4$.

⟨the, ::=n d, ϵ⟩	⟨mouse, ::n, mouse⟩
⟨rat, ::n, rat⟩	⟨rats, ::n, rats⟩
⟨cheese, ::n, cheese⟩	⟨carrot, ::n, carrot⟩
⟨eats, ::=n =d c, λx.λy.eat(x)(y)⟩	⟨eat, ::=n =d c, λx.λy.eat(x)(y)⟩

TABLE 11 | Revised minimalist lexicon X_{41}.

⟨the, ::=num d, ϵ⟩	⟨mouse, ::n -a, mouse⟩
⟨rat, ::n -a, rat⟩	⟨ϵ, ::=n +a num, ϵ⟩
⟨-s, ::=n +a num, ϵ⟩	⟨cheese, ::n, cheese⟩
⟨carrot, ::n, carrot⟩	⟨eats, ::=n =d c, λx.λy.eat(x)(y)⟩
⟨eat, ::=n =d c, λx.λy.eat(x)(y)⟩	

$$\frac{\langle \texttt{the}, ::\texttt{=n d}, \epsilon\rangle \quad \langle \texttt{rat}, ::\texttt{n}, \texttt{rat}\rangle}{\langle \texttt{the rat}, :\texttt{d}, \texttt{rat}\rangle}\ \text{merge-1} \tag{28-1}$$

$$\frac{\langle \texttt{eats}, ::\texttt{=n =d c}, \lambda x.\lambda y.\texttt{eat}(x)(y)\rangle \quad \langle \texttt{carrot}, ::\texttt{n}, \texttt{carrot}\rangle}{\langle \texttt{eats carrot}, :\texttt{=d c}, (\lambda x.\lambda y.\texttt{eat}(x)(y))(\texttt{carrot})\rangle}\ \text{merge-1} \tag{28-2}$$

$$\frac{\langle \texttt{eats carrot}, :\texttt{=d c}, (\lambda x.\lambda y.\texttt{eat}(x)(y))(\texttt{carrot})\rangle}{\langle \texttt{eats carrot}, :\texttt{=d c}, \lambda y.\texttt{eat}(\texttt{carrot})(y)\rangle}\ \lambda\text{-appl.} \tag{28-3}$$

$$\frac{\langle \texttt{eats carrot}, :\texttt{=d c}, \lambda y.\texttt{eat}(\texttt{carrot})(y)\rangle \quad \langle \texttt{the rat}, :\texttt{d}, \texttt{rat}\rangle}{\langle \texttt{the rat eats carrot}, :\texttt{c}, (\lambda y.\texttt{eat}(\texttt{carrot})(y))(\texttt{rat})\rangle}\ \text{merge-2} \tag{28-4}$$

$$\frac{\langle \texttt{the rat eats carrot}, :\texttt{c}, (\lambda y.\texttt{eat}(\texttt{carrot})(y))(\texttt{rat})\rangle}{\langle \texttt{the rat eats carrot}, :\texttt{c}, \texttt{eat}(\texttt{carrot})(\texttt{rat})\rangle}\ \lambda\text{-appl.} \tag{28-5}$$

6.4. Fourth Iteration

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

$$u_4 = \langle \texttt{the rats eat cheese}, \texttt{eat}(\texttt{cheese})(\texttt{rats})\rangle \tag{29}$$

that is unified with the previous lexicon X_3 through our pattern matching algorithm to yield X_4 in **Table 10** in the first place.

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

Next, let us assume that $\mathfrak{L}$ would express the meaning eat(carrot)(rats). It could then attempt the following derivation (30).

$$\frac{\langle \texttt{the}, ::\texttt{=n d}, \epsilon\rangle \quad \langle \texttt{rats}, ::\texttt{n}, \texttt{rats}\rangle}{\langle \texttt{the rats}, :\texttt{d}, \texttt{rats}\rangle}\ \text{merge-1} \tag{30-1}$$

$$\frac{\langle \texttt{eats}, ::\texttt{=n =d c}, \lambda x.\lambda y.\texttt{eat}(x)(y)\rangle \quad \langle \texttt{carrot}, ::\texttt{n}, \texttt{carrot}\rangle}{\langle \texttt{eats carrot}, :\texttt{=d c}, (\lambda x.\lambda y.\texttt{eat}(x)(y))(\texttt{carrot})\rangle}\ \text{merge-1} \tag{30-2}$$

$$\frac{\langle \texttt{eats carrot}, :\texttt{=d c}, (\lambda x.\lambda y.\texttt{eat}(x)(y))(\texttt{carrot})\rangle}{\langle \texttt{eats carrot}, :\texttt{=d c}, \lambda y.\texttt{eat}(\texttt{carrot})(y)\rangle}\ \lambda\text{-appl.} \tag{30-3}$$

$$\frac{\langle \texttt{eats carrot}, :\texttt{=d c}, \lambda y.\texttt{eat}(\texttt{carrot})(y)\rangle \quad \langle \texttt{the rats}, :\texttt{d}, \texttt{rats}\rangle}{\langle \texttt{the rats eats carrot}, :\texttt{c}, (\lambda y.\texttt{eat}(\texttt{carrot})(y))(\texttt{rats})\rangle}\ \text{merge-2} \tag{30-4}$$

$$\frac{\langle \texttt{the rats eats carrot}, :\texttt{c}, (\lambda y.\texttt{eat}(\texttt{carrot})(y))(\texttt{rats})\rangle}{\langle \texttt{the rats eats carrot}, :\texttt{c}, \texttt{eat}(\texttt{carrot})(\texttt{rats})\rangle}\ \lambda\text{-appl.} \tag{30-5}$$

However, uttering the rats eats carrot will probably be rejected by the teacher $\mathfrak{T}$ because of the grammatical number agreement error, thus causing punishment by $\mathfrak{T}$. As a consequence, $\mathfrak{L}$ has to find a suitable revision of its lexicon X_4 that is guided by the underlined matches in **Table 10**.

To this aim, the agent first modifies X_4 as given in **Table 11**.

In **Table 11** 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 ⟨ϵ, ::=n +a num, ϵ⟩ selecting a noun =n and licensing number movement +a, and an item for the plural suffix ⟨-s, ::=n +a num, ϵ⟩ 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 $\mathfrak{T}$. 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 (**Table 2**) above. For now, however, we leave this important problem for future research.

7. DISCUSSION

With the present study we have continued our work on language aquisition and on the unified description of physical interaction and linguistic communication of cognitive agents (Römer et al., 2019; beim Graben et al., 2020). The requirements for a unified description are primarily given by cognitive architectures. That includes the availability of suitable *representational data structures*, the satisfiability of the *composition-, adaptation- and classification principles* as well as the capability for *logical reasoning and learning*. Such an architecture will be particularly useful if it can explain the behavior of cognitive agents as well as the phylogenetic and ontogenetic development of language from earlier stages of evolution without language. Hence, the agent should be based on a physical symbol system (PSS) (Newell and Simon, 1976), that takes physical symbols from its sensory equipment, composing them into symbolic structures (expressions) and transforms them to new expressions that can generate goal directed actions. For this purpose we devised a grammar-based transformation mechanism that unifies physical interaction and linguistic communication using minimalist grammars (MG) and lambda calculus. To explain how the mechanism works, we have selected some example scenarios of the well known mouse-maze problem (Shannon, 1953). With this mechanism, the incoming sensory information from both cognitive loops can be brought together and transformed into meaningful information that can be saved now in a knowledge base and processed logically. The truth-functional approach that is required for the acquisition of a veridical model of a shared environment is dependent on this mechanism. Further, based on the uniform linguistic processing of interaction and communication the communication participants are able to exchange and synchronize their ideas about a common environment. The

use of the information arising from both cognitive loops is also useful here, since linguistic ambiguities can be resolved in this way.

Additionally, 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 (Church, 1936). For the description of syntactic categories we use Stabler's minimalist grammar (Stabler, 1997; Stabler and Keenan, 2003), (MG) a powerful computational implementation of Chomsky's recent Minimalist Program for generative linguistics (Chomsky, 1995). Despite the controversy between Chomsky and Skinner (Chomsky, 1995), we exploit reinforcement learning (Skinner, 2015; Sutton and Barto, 2018) as training paradigm. Since MG codifies universal linguistic competence through the five inference rules (10–11), 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 (Kobele et al., 2002; Stabler et al., 2003; Klein and Manning, 2004; Boston et al., 2010) by positive examples, which is supported by psycholinguistic findings on early human language acquisition (Pinker, 1995; Ellis, 2006; Tomasello, 2006). However, as Pinker (1995) 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 (Moerk, 1983; Sundberg et al., 1996), could effectively 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 of Klein and Manning (2004) 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 (Young, 2010; Baranyi et al., 2015).

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

FUNDING

This work was partly funded by the Federal Ministry of Education and Research in Germany under the grant no. 03IHS022A.

ACKNOWLEDGMENTS

This work was partly based on former joint publications of the authors with Werner Meyer, Günther Wirsching, and Ingo Schmitt (Wolff et al., 2018; beim Graben et al., 2019b, 2020; Römer et al., 2019). The content of this manuscript has been presented in part at the 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2019, Naples, Italy) (beim Graben et al., 2019b; Römer et al., 2019). According to the IEEE author guidelines a preprint of beim Graben et al. (2020) has been uploaded to arXiv.org.

REFERENCES

Artzi, Y., and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Trans. Assoc. Comput. Linguist*. 1, 49–62. doi: 10.1162/tacl_a_00209

Baranyi, P., Csapo, A., and Sallai, G. (2015). *Cognitive Infocommunications (CogInfoCom)*. Springer. ISBN 9783319196077.

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

beim Graben, P., Römer, R., Meyer, W., Huber, M., and Wolff, M. (2019b). "Reinforcement learning of minimalist numeral grammar," in *Proceedings of the 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (Naples: IEEE), 67–72.

beim Graben, P., Römer, R., Meyer, W., Huber, M., and Wolff, M. (2020). Reinforcement learning of minimalist grammars. *arXiv[Preprint].arXiv:2005.00359*.

Bischof, N. (2009). *Psychologie, ein Grundkurs for Anspruchsvolle, 2nd Edn*. Stuttgart: Verlag Kohlhammer.

Bonato, R., and Retoré, C. (2001). "Learning rigid Lambek grammars and minimalist grammars from structured sentences," in *Proceedings of the Third Workshop on Learning Language in Logic* (Strasbourg), 23–34.

Boston, M. F., Hale, J. T., and Kuhlmann, M. (2010). "Dependency structures derived from minimalist grammars," in *The Mathematics of Language, Vol. 6149 of Lecture Notes in Computer Science*, eds C. Ebert, G. Jäger and J. Michaelis (Berlin: Springer), 1–12.

Boullier, P. (2005). "Range concatenation grammars," in *New Developments in Parsing Technology, volume 23 of Text, Speech and Language Technology*, eds H. Bunt, J. Carroll, and G. Satta (Berlin: Springer), 269–289.

Chen, P. P. S. (1975). "The entity-relationship model: toward a unified view of data," in *Proceedings of the 1st International Conference on Very Large Data Bases (VLDB75)*, ed D. S. Kerr (New York, NY: ACM Press), 173.

Chomsky, N. (1995). *The Minimalist Program. Number 28 in Current Studies in Linguistics, 3rd printing 1997*. Cambridge, MA: MIT Press.

Church, A. (1936). An unsolvable problem of elementary number theory. *Am. J. Math*. 58, 345. doi: 10.2307/2371045

Diessel, H. (2013). "Construction grammar and first language acquisition," in *The Oxford Handbook of Construction Grammar*, eds T. Hoffmann and G. Trousdale (Oxford: Oxford University Press), 347–364.

Eliasmith, C. (2013). *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford: Oxford University Press.

Ellis, N. C. (2006). Language acquisition as rational contingency learning. *Appl. Linguist.* 27, 1–24. doi: 10.1093/applin/ami038

Funke, J. (ed.). (2006). *Denken und Problemlsen, Enzyklopdie der Psychologie, 8th Edn*. Hogrefe: Verlag for Psychologie.

Gee, J. P. (1994). First language acquisition as a guide for theories of learning and pedagogy. *Linguist. Educ.* 6, 331–354. doi: 10.1016/0898-5898(94)90002-7

Gold, E. M. (1967). Language identification in the limit. *Inform. Control* 10, 447–474. doi: 10.1016/S0019-9958(67)91165-5

Haegeman, L. (1994). *Introduction to Goverment Binding Theory, volume 1 of Blackwell Textbooks in Linguistics, 2nd Edn, 1st Edn 1991*. Oxford: Blackwell Publishers.

Hale, J. T. (2011). What a rational parser would do. *Cogn. Sci.* 35, 399–443. doi: 10.1111/j.1551-6709.2010.01145.x

Harkema, H. (2001). *Parsing Minimalist Languages* (Ph.D. thesis), University of California, Los Angeles, CA.

Harnad, S. (1990). The symbol grounding problem. *Physica D* 42, 335–346. doi: 10.1016/0167-2789(90)90087-6

Hausser, R. (2014). *Foundations of Computational Linguistics*. Berlin; Heidelberg; New York, NY: Springer Verlag.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree adjunct grammars. *J. Comput. Syst. Sci.* 10, 136–163. doi: 10.1016/S0022-0000(75)80019-5

Joshi, A. K., Vijay-Shanker, K., and Weir, D. (1990). *The convergence of mildly context-sensitive grammar formalisms*. Technical Report MS-CIS-90-01, University of Pennsylvania.

Jungclaussen, H. (2001). *Kausale Informatik*. Wiesbaden: Deutscher Universittsverlag.

Klein, D., and Manning, C. D. (2004). "Corpus-based induction of syntactic structure: models of dependency and constituency," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, eds D. Scott, W. Daelemans, and M. A. Walker (Stroudsburg, PA: Association for Computational Linguistics), 478–485.

Kobele, G. M. (2009). "Syntax and semantics in minimalist grammars," in *Proceedings of ESSLLI 2009* (Bordeaux).

Kobele, G. M., Collier, T. C., Taylor, C. E., and Stabler, E. P. (2002). "Learning mirror theory," in *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+2002)*, ed R. Frank (Stroudsburg, PA: Association for Computational Linguistics), 66–73.

Kracht, M. (2003). *The Mathematics of Language*. Berlin; New York, NY: Mouton de Gruyter.

Kuhlmann, M., Koller, A., and Satta, G. (2015). Lexicalization and generative power in CCG. *Comput. Linguist.* 41, 215–247. doi: 10.1162/COLI_a_00219

Kwiatkowski, T., Goldwater, S., Zettlemoyer, L., and Steedman, M. (2012). "A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL '12* (Stroudsburg, PA: Association for Computational Linguistics), 234–244.

Lausen, G. (2005). *Datenbanken, Grundlagen und XML-Technologien*. München: Elsevier GmbH, Spektrum Akademischer Verlag. 1. Auflage.

Lohnstein, H. (2011). *Formale Semantik und natrliche Sprache, 2nd Edn*. Berlin: De Gruyter.

Mainguy, T. (2010). A probabilistic top-down parser for minimalist grammars. *ArXiv, abs/1010.1826v11*.

Michaelis, J. (2001). "Derivational minimalism is mildly context-sensitive," in *Logical Aspects of Computational Linguistics, Volume 2014 of Lecture Notes in Artificial Intelligence* (Berlin: Springer), 179–198.

Moerk, E. L. (1983). A behavioral analysis of controversial topics in first language acquisition: reinforcements, corrections, modeling, input frequencies, and the three-term contingency pattern. *J. Psycholinguist. Res.* 12, 129–155. doi: 10.1007/BF01067408

Newell, A., and Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Commun. ACM* 19, 113–126. doi: 10.1145/360018.360022

Nivre, J. (2003). "An efficient algorithm for projective dependency parsing," in *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT2003)* (Nancy), 149–160.

Niyogi, S. (2001). "A minimalist implementation of verb subcategorization," in *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT2001)* (Beijing).

Pinker, S. (1995). "Language acquisition," in *Language: An Invitation to Cognitive Science, Chapter 6*, eds L. R. Gleitman, D. N. Osherson, M. Liberman, L. R. Gleitman, D. N. Osherson, and M. Liberman (Cambridge, MA: MIT Press), 135–182.

Römer, R., beim Graben, P., Huber, M., Wolff, M., Wirsching, G., and Schmitt, I. (2019). "Behavioral control of cognitive agents using database semantics and minimalist grammars," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (Naples).

Schönfinkel, M. (1924). Über die Bausteine der mathematischen Logik. *Mathematische Annalen* 92, 305–316. doi: 10.1007/BF01448013

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theor. Comput. Sci.* 88, 191–229. doi: 10.1016/0304-3975(91)90374-B

Shannon, C. E. (1953). Computers and automata. *Proc. Institute Radio Eng.* 41, 1234–1241. doi: 10.1109/JRPROC.1953.274273

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

Stabler, E. (1997). "Derivational minimalism," in *Logical Aspects of Computational Linguistics (LACL96), volume 1328 of Lecture Notes in Computer Science*, ed C. Retoré (New York, NY: Springer), 68–95.

Stabler, E. P. (2011a). "Computational perspectives on minimalism," in *Oxford Handbook of Linguistic Minimalism*, ed C. Boeckx (Oxford: Oxford University Press), 617–641.

Stabler, E. P. (2011b). "Top-down recognizers for MCFGs and MGs," in *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics* (Portland, OR: Association for Computational Linguistics), 39–48.

Stabler, E. P., Collier, T. C., Kobele, G. M., Lee, Y., Lin, Y., Riggle, J., et al. (2003). "The learning and emergence of mildly context sensitive languages," in *Advances in Artificial Life, volume 2801 of Lecture Notes in Computer Science*, eds W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler (Berlin: Springer), 525–534.

Stabler, E. P., and Keenan, E. L. (2003). Structural similarity within and among languages. *Theor. Comput. Sci.* 293, 345–363. doi: 10.1016/S0304-3975(01)00351-6

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

Sundberg, M. L., Michael, J., Partington, J. W., and Sundberg, C. A. (1996). The role of automatic reinforcement in early language acquisition. *Anal. Verbal Behav.* 13, 21–37. doi: 10.1007/BF03392904

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA; London: The MIT Press.

Tomasello, M. (2006). First steps toward a usage-based theory of language acquisition. *Cogn. Linguist.* 11:61. doi: 10.1515/cogl.2001.012

van Zaanen, M. (2001). "Bootstrapping syntax and recursion using alignment-based learning," in *Proceedings of the Seventeenth International Conference on Machine Learning* (San Francisco, CA), 1063–1070.

Vera, A., and Simon, H. (1993). Situated action: a symbolic interpretation. *Cogn. Sci.* 17, 7–48. doi: 10.1207/s15516709cog1701_2

Wegner, P. (2003). *Lambda Calculus*. Hoboken, NJ: John Wiley and Sons Ltd., GBR.

Wirsching, G., and Lorenz, R. (2013). "Towards meaning-oriented language modeling," *in Proceedings of the 4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (Budapest), 369–374.

Wolff, M., Huber, M., Wirsching, G., Römer, R., Graben, P., Schmitt, I. (2018). "Towards a quantum mechanical model of the inner stage of cognitive agents," in *2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (Budapest: IEEE).

Wolff, M., Römer, R., and Wirsching, G. (2015). "Towards coping and imagination for cognitive agents," in *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)* (Gyor: IEEE), 307–312.

Wolff, M., Tschpe, C., Römer, R., and Wirsching, G. (2013). "Subsymbol-symbol-transduktoren," in *Proceedings of "Elektronische Sprachsignalverarbeitung*

(ESSV)," volume 65 of Studientexte zur Sprachkommunikation, eds P. Wagner (Dresden: TUDpress), 197–204.

Young, S. (2010). "Still talking to machines (cognitively speaking)," in *Conference of the International Speech Communication Association, INTERSPEECH 2010*, eds T. Kobayashi, K. Hirose, and S. Nakamura (Makuhari: ISCA), 1–10.

Zettlemoyer, L. S., and Collins, M. (2005). "Learning to map sentences to logical form: structured classification with probabilistic categorial grammars," in *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence, UAI'05* (Arlington, VA: AUAI Press), 658–666.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.